VOLUM I

Memòria

Deployment de Spark al Supercomputador MareNostrum III

Albert Calvo Ibáñez

27/04/2016

Director: Jordi Torres Viñals (Arquitectura de Computadors)

Codirector: Rubèn Tous Liesa (Arquitectura de Computadors)

Grau en Enginyeria Informàtica

Especialitat en Tecnologies de la Informació

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) – BarcelonaTech

Aquest treball de final de grau tracta un tema d'actualitat i de gran interès. En concret es fa una introducció al món de l'anàlisi de dades massives utilitzant l'eina Spark. Durant el desenvolupament d'aquest projecte es fa una comparació entre diferents plataformes que permeten treballar amb Spark. Es treballa amb la plataforma MareNostrum III, Supercomputador del Barcelona Supercomputing Center, amb el Centre de Càlcul del Departament de Computadors i, finalment, amb la plataforma en línia Databricks.

A partir del coneixement adquirit treballant i testejant amb les plataformes, es crearà material didàctic per a l'assignatura Supercomputers Architecture (SA) del Master in Research Informatics (MIRI) i per a l'assignatura Cloud Computing (CC) del Màster en Enginyeria Informàtica, que pertanyen a la Facultat d'Informàtica de Barcelona (FIB).

This Final Degree Project is about a topic of current interest. In particular, this project is an introduction to the world of massive data analysis using Spark. During this project it's made a comparison between different platforms that can work with Spark, MareNostrum III supercomputer of the Barcelona Supercomputing Center, the Computer of the Department of Computer Architecture and, finally, with the online platform Databricks .

From the previous knowledge acquired, working with the different platforms, it's made teaching material for the Supercomputers Architecture course (SA) of the Master in Research Informatics (MIRI) and for the Cloud Computing course (CC) of the Engineering Computer Master (MEI). Both Master belong to the Barcelona Informatics Faculty (FIB).

0. Índex

1. Introducció i abast

Aquest treball de final de grau es desenvolupa al Departament d'Arquitectura de Computadors de la Facultat d'Informàtica de Barcelona. El projecte compta amb la col·laboració del Barcelona Supercomputing Center.

1.1 Introducció

El projecte abasta temes d'actualitat i, específicament, tracta l'anàlisi de dades massives utilitzant una plataforma HPC (High Performance Computing) [1] com és el supercomputador MareNostrum III . El projecte comptarà amb dos vessants ben diferents: un de tècnic, en què es buscarà d'entendre les eines que existeixen i es passarà a treballar-hi. Concretament, es durà a terme una comparació entre diferents infraestructures que treballen amb Spark, i un altre de divulgatiu, en què es crearà material per a incentivar l'estudi d'aquest àmbit, que actualment es troba en expansió.

En aquest document es plasma tot el treball realitzat. En els dos primers capítols es fa una introducció al projecte, s'hi inclou una introducció a com s'ha dut a terme la gestió del projecte: Introducció, estat de l'art, definició de l'abast, planificació temporal, recursos, valoració d'alternatives i pla d'acció, gestió econòmica i l'informe de sostenibilitat. També s'hi inclou una àmplia introducció a Spark que té com a objectiu divulgar els conceptes bàsics d'aquest llenguatge i que siguin d'ajuda al lector per entendre els capítols següents de caire mes tècnic.

Als capítols 3 i 4, s'hi explica com s'han desenvolupat els diferents experiments que es duen a terme amb la finalitat de comparar les infraestructures i, al capítol 4, s'hi exposen els resultats de les proves que s'han fet. Els capítols 5 i 6 són una ampliació dels experiments.

Tal i com s'ha dit anteriorment, aquest treball té un vessant didàctic. Al capítol 7, s'hi explica el material didàctic desenvolupat.

Finalment, l'últim capítol és dedicat a les conclusions, on es valora si s'han acomplert els diferents objectius del projecte, les diferents conclusions fruit del desenvolupament del projecte i les variacions en la planificació i pressupost plantejades a l'inici del document.

1.1.2 Objecte d'estudi

L'objecte d'estudi d'aquest projecte se centra en l'anàlisi massiva de dades, també coneguda com a Big Data [2]. Aquesta anàlisi permet de ser aplicada en molts àmbits diferents, des d'aplicacions sanitàries, socials i econòmiques, fins a aplicacions d'oci o recreatives.

Per tant, aquest treball busca de conèixer quines en són les eines i les metodologies que existeixen avui en dia. Per tal de poder fer aquesta anàlisi, s'estudiarà en profunditat el concepte Big Data que, tot i la popularitat que ha obtingut en els darrers anys, pot causar confusió. A més, també s'analitzaran algunes de les tècniques que trobem per a dur a terme l'anàlisi i com treballen a la plataforma MareNostrum III. Per acabar, s'estudiarà i es compararà amb una plataforma alternativa.

1.1.3 Objectius del projecte

A continuació, es llisten els diferents objectius que té associats aquest projecte. Al final s'hi debatran quins objectius s'han desenvolupat i en quina mesura:

- Realitzar proves de rendiment a la plataforma MareNostrum III per a l'anàlisi Big Data en què es buscaran de quines variables depenen les execucions, quines es podrien millorar i la seva repercussió en el sistema.

- Comparar la plataforma MareNostrum III amb plataformes cloud (que explicaré més endavant). Aquesta comparativa permetrà estudiar les diferències d'execució entre plataformes.

- Centralitzar les diferents proves que s'han dut a terme fins ara al Supercomputador i crear material (majoritàriament en forma de hands-on) que permeti els estudiants i investigadors d'aprendre a executar Spark de manera senzilla. El pròxim quadrimestre s'utilitzaran els hands-on desenvolupats en les assignatures Supercomputers Architecture i Cloud Computing dels màsters de la FIB: Master in Research Informatics (MIRI) i Màster en Enginyeria Informàtica (MEI).

1.1.4 Metodologia

La metodologia del treball és l'ús de cicles curts. Aquesta metodologia es basa en fites a acomplir en un temps determinat. Més endavant es fixaran uns objectius que s'hauran d'assolir en el temps

estipulat en el document. Cada fita serà presentada al director/codirector del projecte, el qual validarà que cada una d'elles s'hagi assolit.

Durant tot el projecte s'utilitza el framework Spark, que serà explicat més endavant. Aquest framework es programa amb Scala, un llenguatge de programació funcional. S'utilitzarà també l'eina Maven, un organitzador de projectes que permet estructurar les diferents parts, resoldre les dependències i compilar els binaris.

El Supercomputador MareNostrum III disposa de SSH (Secure Shell) per tal de connectar-s'hi. Per tant, totes les proves es faran des d'un ordinador personal. Amb la finalitat de poder utilitzar Spark, s'emprarà l'eina SPARK4MN [3] que permet carregar l'entorn per treballar amb Spark.

Addicionalment, s'utilitzarà la plataforma Bluemix que ofereix Spark com a servei. Bàsicament, es dedica a fer que els clients tinguin una interfície web on insereixen dades i que Bluemix, mitjançant Spark, processi les dades i en mostri el resultat al client. Posteriorment es cobrarà en concepte d'hores de càlcul. Per accedir en aquesta plataforma s'utilitzarà un ordinador personal i, en aquest cas, un navegador web.

Durant el projecte, s'ha establert un seguit d'eines de seguiment. Aquestes eines permeten que el director del projecte i codirector puguin conèixer en qualsevol moment quin és l'estat del projecte. Els diferents hands-on i documents de text realitzats seran compartits mitjançant la plataforma Google Drive. Tot el codi que es faci serà compartit mitjançant Github que és una plataforma de desenvolupament col·laboratori, és a dir, tot el codi del meu projecte pot ser vist per qualsevol persona, la qual pot modificar-lo i clonar-lo a través del seu compte.

La planificació es farà a partir d'un diagrama de Gantt (es veurà més endavant). Al principi de cada tasca (aproximadament cada dues setmanes) es durà a terme una reunió amb el director del projecte (de manera presencial o bé per *email*) on s'exposarà si les tasques anteriors han estat assolides amb èxit.

Tot el projecte es farà, sempre que sigui possible, amb eines de lliure distribució. És important destacar que tant Spark, Maven com Scala tenen llicència Creative Commons.

1.1.5 Actors implicats

El públic a qui va dirigit aquest projecte essencialment són estudiants que volen aprendre a utilitzar la plataforma. El material creat tindrà un nivell que sigui útil per a estudiants d'informàtica, que

tinguin nocions de programació i que vulguin endinsar-se en aquesta temàtica.

Per una altra banda, també serà útil per a investigadors, tant els propis del BSC (Barcelona Supercomputer Center) com externs. Les proves que es facin al supercomputador permetran de conèixer com funciona i es comporta la plataforma, cosa que pot ajudar d'altres centres de recerca a esbrinar si els resultats que obtenen són similars a la infraestructura del BSC.

1.2 Estat de l'art

Actualment l'anàlisi de dades té una multitud d'aplicacions, és a dir, existeix un gran nombre de solucions per a aquesta demanda creixent, sobretot per a la sortida al mercat dels dispositius IoT (Internet of Things) [4]. Aquests dispositius són de baix cost, estan connectats a la xarxa i emeten dades que calen ser analitzades i tractades. En aquest punt és quan la tecnologia d'anàlisi massiva permet trobar correlacions entre la informació recol·lectada pels dispositius. Per tant, la demanda de l'anàlisi és molt alta i són diverses les solucions que existeixen avui en dia. Aquest apart queda dividit en dos subapartats. En el primer, s'hi defineixen i s'hi expliquen les característiques del Big Data i, en el segon subapartat, s'hi expliquen algunes de les eines per tractar amb dades massives.

1.2.1 Big Data

Segons la consultora Gartner, el terme de «Big Data» es defineix de la manera següent:

> *"Big Data is a high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation."*[5]

L'anterior definició es pot traduir de la manera següent: *El Big Data és un seguit d'activitats de gran quantitat de dades, de gran velocitat i/o gran varietat d'informació que exigeixen formes rendibles i innovadores de processament d'informació que milloren la presa de decisions i l'automatització de processos.*

Les característiques del Big Data es defineixen en les conegudes 4V del Big Data [6]. Aquestes són: volum, velocitat, varietat i veracitat.

- Volum: El volum fa referència a la mida del conjunt de dades amb què compta el projecte.

Quan es compta amb poques dades es parla de Mineria de Dades o bé de Business Intelligence[1]. Aquesta classificació és bastant subjectiva, ja que un projecte pot comptar amb un conjunt de dades de 10 Terabytes i ser considerat Big Data, i, d'altra banda, una empresa més petita que compti amb un conjunt de dades de 10 GB també seria considerada Big Data.

- Velocitat: Tal i com hem vist anteriorment, es treballa amb grans quantitats de dades que per analitzar-les calen grans recursos, a diferència del Bussines Intelligence en què els càlculs i l'anàlisi es fa en batch[2]. En el Big Data cal aconseguir resultats el més aviat possible, per tant, calen grans Centres de Càlcul per aconseguir fer l'anàlisi de dades en curts períodes de temps o bé en temps real[7].
- Varietat: L'erupció d'Internet de les coses (de l'anglès: Internet of Things[3]) i de les xarxes socials, crea de manera diària tones d'informació que pot ser analitzada. Aquestes dades poden venir de diferents fonts. Les fonts es classifiquen en dos tipus: Fonts estructurades (per exemple, dades provinents de bases de dades relacionals, fulls de càlcul, mesures d'un sensor, etc.); fonts de dades no estructurals (per exemple, les que provenen d'imatges, vídeos, piulades de Twitter, etc.).
- Veracitat: aquesta característica quantifica la qualitat de les fonts de les dades. Les dades han de provenir de fonts fiables.

1.2.2 Eines

Actualment existeixen multituds d'eines que permeten treballar amb el Big Data. A la **figura 1** es pot observar un diagrama amb les eines que existeixen. El diagrama es va fer l'any 2014. Atès que el camp del Big Data està en constant creixement, el més probable és que, ara per ara, es puguin afegir noves eines en aquest diagrama.

1 Bussiness Intelligence: Intel·ligència de negocis. Aquest terme fa referència a l'estratègia empresarial que busca incrementar el rendiment, l'eficiència i la competitivitat del negoci mitjançant l'anàlisi de dades històriques de l'empresa [2.6]

2 Process Batch: Execució de processos sense supervisió humana. Es processa gran quantitat de dades en grans períodes de temps.

3 Internet of Things: Xarxa formada per un conjunt d'objectes connectats a Internet que es poden comunicar entre ells i també amb humans i, d'aquesta manera, transmetre i tractar dades amb intervenció humana o sense. Definició extreta de http://www.termcat.cat

Figura 1: Big Data Landscape

Font:datafloq.com

Una d'aquestes solucions, i una de les més conegudes, és Hadoop. Hadoop va ser creat pel nord-americà Doug Curring l'any 2009 i va ser creat mitjançant un article que va publicar Google on s'explicava la tecnologia MapReduce [8] que empraven a la companyia. Doug Curring va crear una eina semblant, però de lliure distribució, anomenada Hadoop. Es tracta d'un framework que permet treballar en un alt nivell en sistemes distribuïts. Per tant, el programador no ha de preocupar-se de com el sistema distribueix internament la feina en el clúster.

Hadoop compta amb: HDFS (Hadoop Distributed File System) que permet la distribució de la informació als diferents nodes; Hadoop Yarn que permet programar tasques; Hadoop MapReduce que permet processar datasets en paral·lel.

Una de les alternatives al framework Hadoop és el framework Spark. El projecte Apache Spark neix l'any 2010 a Berkeley i l'any 2013 és lliurat a la fundació Apache Software Foundation que el publica amb llicència Apache 2.0. Des de l'any 2013 fins a l'actualitat, el projecte ha anat madurant i, a hores d'ara, en són més de 460 els contribuïdors.

Un cop vistes les característiques que perfilen Spark, l'ús d'aquesta eina és cada vegada més

utilitzada i el Supercomputador MareNostrum III també l'ha començat a utilitzar com a framework Big Data. Actualment, s'hi poden executar tasques però no se n'aconsegueix una escalabilitat perfecta. L'escalabilitat perfecta és aquella en què incrementant el nombre de recursos es redueix el temps del problema de manera proporcional. Millorant aspectes de cada execució: paràmetres interns de Spark o bé paràmetres propis de la màquina, com la tipologia de xarxa que cal utilitzar o el nombre total de fils d'execució, podem aconseguir millorar-ne el rendiment. Aquests ajustaments tenen com a objectiu dur a terme les tasques en un temps menor i utilitzar-hi el menor nombre de recursos.

Diversos investigadors del BSC han fet proves de rendiment que perfilen com són els temps i els resultats d'execucions al MareNostrum III. Aquestes proves s'han dut a terme amb l'eina de benchmarking bsc.spark (SPARK4MN) que permet executar càrregues de treball per analitzar-ne el rendiment.

1.3 Definició de l'abast

1.3.1 Abast

Atès que sóc novell en les tecnologies que es tractaran en aquest projecte, primer de tot caldrà entendre el mecanisme de les eines amb les quals es treballarà. Aquest estudi es basarà en el fet d'entendre com funciona la plataforma ja creada SPARK4MN i l'eina de benchmarking bsc.spark disponible a Github. En aquesta etapa del projecte es crearà un document que expliqui de manera detallada quins són els passos per fer la connexió al Supercomputador MareNostrum III i executar-ne un primer programa de prova.

Un cop entès com funciona la plataforma, es passarà a analitzar-la tot executant diferents càrregues de treball que, prèviament seran analitzades i acceptades per algun expert en el tema. Aquestes càrregues de treball es duran a terme mitjançant l'eina bsc.spark. Una vegada aconseguits els resultats de les proves es passaran a analitzar a partir d'un punt crític i es crearà un document tècnic que serveixi de guia per conèixer-ne el rendiment. Aquestes proves es faran amb l'execució de l'algorisme k-means [9]. Seguidament, es passarà a analitzar una plataforma alternativa a la pròpia del BSC,

1.3.2 Obstacles

Aquest projecte compta amb diversos obstacles ja previstos, atesa l'experiència d'altres investigadors que anteriorment han executat proves a la màquina.

Un dels obstacles en la programació al Supercomputador MareNostrum III es troba en l'usuari que juga el rol de programador, ja que només té accés per a la programació en un alt nivell i, uns possibles errors que es puguin trobar en un baix nivell, són imperceptibles. En el cas que es tingués accés en aquesta capa més baixa de la màquina se'n podrien optimitzar encara més les execucions.

Aquesta optimització està fora de l'abast d'aquest TFG i, per tant, no serà avaluada. Si es donés la circumstància de detectar algun possible cas en què una anàlisi a baix nivell pogués millorar-ne les execucions, serien explicades per a futurs investigadors que volguessin treballar a l'àrea.

Un altre obstacle són els possibles errors de programació. Atès que són imprevisibles, pot ser que afectin la programació en el calendari i en el temps de realització del projecte. Si em fóra impossible de solucionar-ne l'error, demanaria ajuda a d'altres usuaris que estiguessin utilitzant el mateix framework.

1.3.3 Ampliacions

Com que aquest projecte compta amb unes hores de dedicació i un calendari estricte, si es donés la circumstància que s'hagués d'estendre el projecte es faria amb alguna de les ampliacions següents:

- Fer les proves de rendiment amb altres algorismes. Aquesta nova anàlisi servirà per conèixer el comportament d'algorismes amb els quals encara no se n'han fet proves al supercomputador.

- Estendre el paquet bsc.spark per tal d'encabir nous experiments i funcionalitats.

1.4. Planificació temporal

Aquest document explica la planificació prevista per al projecte i la divisió del projecte en blocs i tasques. El projecte té una durada fixada de 4 mesos, des de setembre de 2015 fins a gener de 2016.

El projecte queda dividit en quatre blocs de treball. En els dos apartats següents es fa un incís per explicar les dues metodologies que es faran servir i, a continuació, es passa a descriure cada un dels blocs del projecte.

Bloc	Descripció	Metodologia
Bloc 0	Familiarització	-
Bloc I	Curs de GEP.	Cascada
Bloc II	Desenvolupament del projecte.	Scrum
Bloc III	Preparació de la defensa.	Cascada

Taula 1: Divisió del projecte en blocs

La càrrega de treball és de 450 hores (equivalent a 18 crèdits ECTS). A la **figura 2**, s'hi pot observar la distribució d'hores de treball.

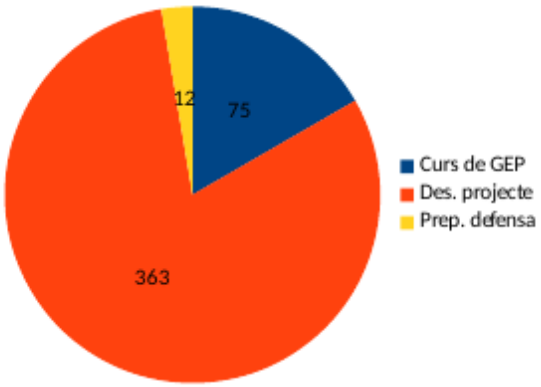

Figura 2: Hores de dedicació total al projecte

Font: Elaboració pròpia

1.4.1 Metodologia en Cascada

Segons aquesta metodologia, cadascuna de les tasques s'executa de manera lineal i, per tant, fins que no s'acaba una tasca no es pot passar a realitzar-ne la següent. La metodologia és utilitzada durant el projecte en els blocs I i III ja que ens trobem davant de tasques senzilles. Per al bloc II, en canvi, s'ha escollit d'utilitzar la metodologia Scrum (explicada més endavant, a l'apartat 1.4.2).

1.4.2 Metodologia Scrum

Sistema de treball

Aquesta metodologia també s'utilitza durant el desenvolupament del projecte. Es tracta d'un mètode àgil que es basa en dur a terme lliuraments incrementals de desenvolupament d'un producte o servei. Una característica que fa atractiva aquesta metodologia és que en poques iteracions es pot obtenir una visió general del projecte.
La metodologia divideix les tasques en històries i, en cada iteració (període curt de temps), se'n desenvolupa una o diverses històries.

Rols

La metodologia defineix els rols següents:
- ProductOwner: és la veu crítica del client. Aquest rol és l'encarregat de validar cada una de les històries. Aquesta tasca serà duta a terme pel director o bé pel codirector del projecte.
- ScrumMaster: l'objectiu d'aquest rol és eliminar els obstacles que impedeixen que s'acompleixi cada una de les històries. Aquesta tasca la farà el director o bé el codirector del projecte.
- Equip de desenvolupament: encarregat de desenvolupar cada una de les iteracions. Aquesta tasca serà duta a terme per l'autor del document.

Reunions i validacions

Al final de cada iteració es validarà si s'han desenvolupat amb èxit les històries. S'enviarà un resum de l'estat al director del projecte indicant l'assoliment de la iteració i una valoració global del projecte.

1.4.3. Blocs

Bloc 0: Familiarització

Un pas previ per poder fer aquest projecte és la familiarització tant amb Scala com amb Spark. Aquesta familiarització es va fer durant el mes de juliol quan també es va estudiar el paper Spark Deployment and Performance Evaluation on the MareNostrum Supercomputer'[10], al qual s'hi expliquen els primers experiments de Spark a la plataforma MareNostrum III. Aquest estudi previ ha servit per entendre quins són els obstacles de treball i l'estat de l'art.

Bloc I: Curs de GEP

Descripció

Aquest bloc correspon al curs de GEP que comença el dia 15 de setembre de 2015 i acaba el 23 d'octubre de 2015. La dedicació és de 75 hores de treball.

A continuació es divideix el curs de GEP en subtasques i la planificació temporal (**taula 2**).

ID.	Descripció	Inici	Final	Durada	Hores de dedicació
Tasca 1	Definició de l'abast i contextualització	15/09/15	22/09/15	6 dies	13 hores
Tasca 2	Planificació temporal	23/09/15	29/09/15	5 dies	11 hores
Tasca 3	Gestió Econòmica i sostenibilitat	29/09/15	06/10/15	5 dies	11 hores
Tasca 4	Presentació preliminar	06/10/15	09/10/15	4 dies	12 hores
Tasca 5	Document final i PPT de la presentació	09/10/15	16/10/15	6 dies	13 hores
Tasca 6	Plec de condicions	16/10/15	23/10/15	6 dies	15 hores

Taula 2: Divisió de tasques del curs de GEP

Diagrama de Gantt

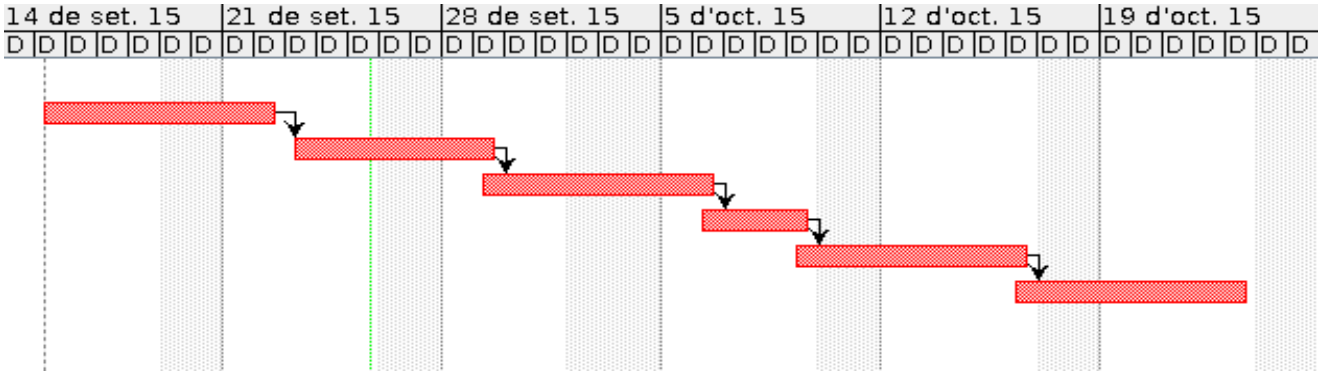

Figura 3: Diagrama de Gantt del curs de GEP

Font: Elaboració pròpia

Bloc II: Desenvolupament del projecte

Descripció

Aquest bloc correspon al cos del projecte. Es tracta del bloc amb més durada i més hores de dedicació. Tenint present la seva complexitat s'ha decidit d'utilitzar la metodologia Scrum. Aquest bloc es desenvolupa a principis de setembre fins a finals de gener de 2016. A continuació es llisten les històries que es desenvoluparan durant el projecte.

-Història 1: Boilerplate Spark - Creació de binaris que contenen l'esquelet d'una aplicació Spark.
-Història 2: Hands-on (introducció) – Introducció a Spark per al MareNostrum III.
-Història 3: Proves amb Spark – Desenvolupament d'experiments al MareNostrum III.
-Història 4: Introducció a la plataforma Cloud – Estudi de la plataforma.
-Història 5: Hands-on (plataforma Cloud*)* – Introducció a la plataforma cloud triada.
-Història 6: Hands-on (cas pràctic) – Execució d'un programa al MareNostrum III.
-Història 7: Comparativa de plataformes - Comparativa entre MareNostrum III i la plataforma Cloud.
-Història 8: Conclusions i resultats – Validació i anàlisi dels resultats.
-Història 9: Ampliacions – Història prevista per ampliacions.

Iteracions i fites

El projecte s'ha dividit en un total de nou iteracions d'onze dies de durada. A continuació es descriu per cadascuna de les iteracions, quines són les seves dates, durada, hores de dedicació (aproximada) i les històries a desenvolupar.
Cada iteració té una durada aproximada de 40 hores. En el cas que en una iteració es completin les històries, es passarà a treballar amb la història següent. També s'hi inclou quina és la data aproximada de cada història.

ID	Inici	Final	Durada	Dedicació	Històries
IT1	15/09/15	29/09/15	11 dies	40 hores	H1, H2
IT2	30/09/15	15/10/15	11 dies	40 hores	H3
IT3	16/10/15	30/10/15	11 dies	40 hores	H1, H2
IT4	20/10/15	10/11/15	11 dies	40 hores	H4, H5
IT5	30/10/15	16/11/15	11 dies	40 hores	H3, H7

IT6	19/11/15	03/12/15	11 dies	40 hores	H6,H3
IT8	04/12/15	18/12/15	11 dies	40 hores	H3, H7
IT8	21/12/15	04/01/16	11 dies	40 hores	H7, H6
IT9	05/01/16	19/01/16	11 dies	40 hores	H8, H9

Taula 3: Calendari d'Iteracions

ID	Iteracions	Hores de dedicació	Final
H1	2	30 hores	30/10/15
H2	2	50 hores	30/10/15
H3	4	135 hores	15/10/15
H4	1	10 hores	26/10/15
H5	1	30 hores	10/11/15
H6	2	30 hores	03/12/15
H7*	2	45 hores	18/12/15
H8	1	20 hores	19/01/16
H9	1	20 hores	19/01/16

Taula 4: Dedicació a cada història

*Es preveu que la història 7 tindrà més dificultat que la preestablerta. En el cas que fos necessari, s'acabarà de desenvolupar durant la història d'ampliació.

Diagrama de Gantt

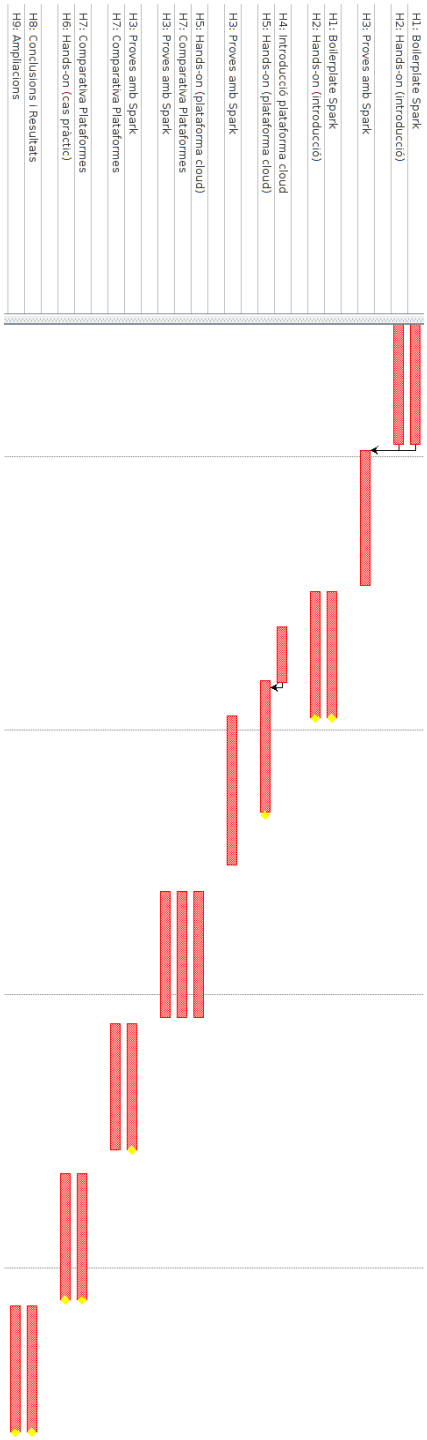

Figura 4: Diagrama de Gantt del desenvolupament del projecte.

Font: Elaboració pròpia

Bloc III: Preparació de la defensa

Descripció

Aquest últim bloc correspon al fet de revisar l'estil de la memòria i preparar-ne la presentació davant del tribunal. El bloc es durà a terme del 5 de gener al 20 de gener.

ID.	Descripció	Inici	Final	Durada	H. dedicació
Tasca 1	Preparació de la defensa	05/01/16	20/01/16	11 dies	20 hores

Taula 5: Divisió de tasques de la preparació de la defensa

Diagrama de Gantt

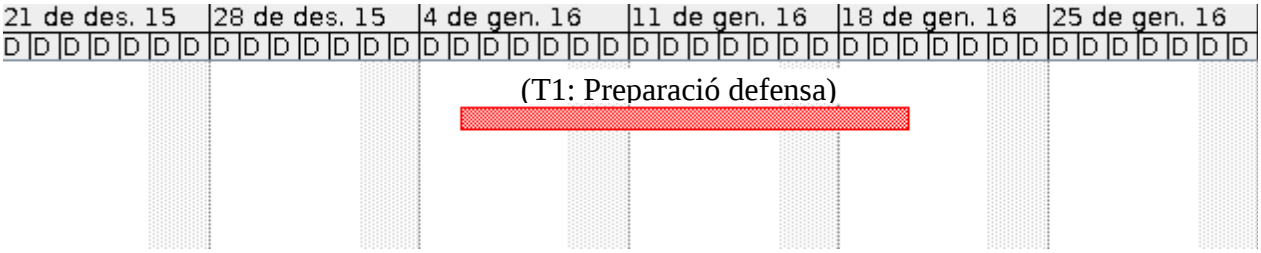

Figura 5: Diagrama de Gantt de la preparació de la defensa

Font: Elaboració pròpia

1.4.4 Diagrama de Gantt complet

El diagrama complet es troba disponible a l'Annex 1. Planificació temporal.

1.5 Recursos

1.5.1 Recursos Humans

A continuació es llisten els recursos humans necessaris per a desenvolupar el projecte.

- Cap de projecte: és la persona responsable de la planificació, execució i control del projecte. També és l'encarregat de coordinar els altres rols del projecte.

- Analista: és la persona encarregada d'estudiar el model i de proposar de quina manera ha d'actuar el programador.

- Programador: és la persona que escriu el codi font del projecte a partir de les especificacions del cap de projecte i analista.

1.5.2 Recursos de maquinari i programari

Durant el projecte s'utilitzaran els segons recursos de maquinari i de programari. Els tres primers recursos són de maquinari i els següents són recursos de programari.

Recurs	Tipus	Descripció
Ordinador Personal – equipat amb un processador Intel quad core (2.16GHz), 4GB de memòria RAM i 500 GB de disc dur	Eina de desenvolupament	Ordinador amb el qual es desenvoluparà tot el projecte
Webcam	Eina de desenvolupament	Permetrà realitzar la tasca de vídeo del curs de GEP
Supercomputador MareNostrum III	Eina de desenvolupament	S'hi accedirà via SSH per a desenvolupar-ne els experiments
Fedora 22	Eina de desenvolupament	Sistema operatiu basat en Linux, que és el sistema utilitzat a l'ordinador personal
LibreOffice	Eina de desenvolupament	Editor de textos, que és l'utilitzat per a la memòria del projecte
Project Libre	Eina de gestió	Eina per a la gestió del temps, que permet crear diagrames Gantt
Atom	Eina de desenvolupament	Editor de textos
Spark	Eina de desenvolupament	Framework Big Data
Scala	Eina de desenvolupament	Llenguatge de programació
Maven	Eina de gestió	Gestor de dependències i generador de paquets
Github	Eina de gestió desenvolupament	Gestor de versions
Bluemix	Eina de desenvolupament	Plataforma Cloud

Taula 6: Recursos del projecte

1.6. Valoració d'alternatives i pla d'acció

El projecte té una durada aproximada de quatre mesos. Per tal d'assegurar la finalització del projecte en el temps preestablert es defineixen les alternatives següents per al bloc II: Desenvolupament del

projecte, en què existeixen històries que poden presentar dificultats.

Història 3: En aquesta història es preveu treballar amb l'algorisme k-means però, en el cas de no poder-hi treballar, es faran proves de rendiment amb altres algorismes.

Història 6: En aquesta història es busca de fer una execució amb dades reals. La dificultat recau en el fet de no trobar dades. En aquest cas es destinaran les hores de la Història 6 a la Història 3.

Història 7: En aquesta història, es preveu treballar amb la plataforma Cloud Bluemix. En el cas que no fos possible es treballaria amb la plataforma cloud alternativa d'Amazon AWS o bé amb Databricks.

Per aplicar una alternativa cal estudiar si la història inicial és impossible de fer-la. Per tal d'assegurar la finalització del projecte, el millor és treballar amb l'alternativa. Aquest canvi es farà de manera pactada amb el director o codirector del projecte.

1.7. Gestió econòmica i sostenibilitat

1.7.1 Gestió econòmica

En aquest apartat es detallen, de manera aproximada, els costos directes (aquells que estan lligats a les tasques del digrama de Gantt, a l'apartat 2.5.). Els dos altres costos no estan lligats directament a tasques del digrama de Gantt, com ara la llum o l'assistència a conferències.

Costos Directes

Els costos directes s'han dividit depenen del tipus de recurs emprat.

Recursos de maquinari

Producte	Unitats	Preu Unitat	Vida útil	Amortització	Preu
Ordinador Personal	1	722€	7 anys	42.97€	42.97€
Imprevist – (risc 5%) Reparació/Ampliació d' ordinador	1	36.1€	-	-	36.1€
Web-Cam	1	20.99€	5 anys	1.74€	1.74€

MareNostrum*	1	-	-	-	
TOTAL					44.71€

Taula 7: Taula de costos directes de maquinari

* El supercomputador MareNostrum és un bé públic, ja que es paga entre tots els contribuents. Per tant, no suposa un cost directe per al projecte. De manera aproximada es poden quantificar aquests recursos en 400 €.

Recursos de programari

Producte	Unitats	Preu Unitats	Vida útil	Amortització	Preu
Fedora 22	1	0€	-	-	0€
LibreOffice	1	0€	-	-	0€
ProjectLibre	1	0€	-	-	0€
Atom	1	0€	-	-	0€
Spark	1	0€			0€
Scala	1	0€	-	-	0€
Maven	1	0€	-	-	0€
Github	1	0€	-	-	0€
Bluemix*	1	0€	-	-	0€
TOTAL					0€

Taula 8: Costos directes de programari

* La plataforma Bluemix és de pagament però IBM n'ha becat l'accés per desenvolupar el projecte. Aquesta beca es pot quantificar, aproximadament, en 750 €.

Recursos Humans

En aquest projecte tots els rols seran desenvolupats per l'autor del document. El director del projecte és el rol encarregat de fer el curs de GEP, escriure'n la memòria i preparar-ne la defensa. Els rols de programador i analista (última tasca del diagrama de Gannt, validar i raonar els resultats) són els encarregats de desenvolupar el projecte.

Producte	Unitats	Preu Unitats	Vida útil	Amortització	Preu
Cap de Projecte	150 h	20 €/h	-	-	3000 €
Programador	220 h	20 €/h	-	-	4400 €
Imprevist – (risc 10%) previsió insuficient *	22 h	20 €/h			440 €
Analista	80 h	25 €/h	-	-	2000 €
TOTAL					9400 €

Taula 9: Costos directes de recursos humans

* El nombre d'hores previstes per al desenvolupament han estat insuficients i, per tant, es defineixen unes hores de més per a la seva finalització.

Costos indirectes

Producte	Unitats	Preu Unitat	Vida útil	Amortització	Preu
Electricitat	450 h	0,045 €/h	-	-	20,43 €
Accés a Internet	450 h	0,032 €/h	-	-	14,4 €

Taula 10: Costos indirectes

Càlcul d'electricitat suposant: 0,1261 €/Kwh i un ordinador amb una potència de 0,36 Kwh.

Altres costos

Durant la durada del projecte s'assistirà als meetup's *(*xarxa social que posa en contacte una comunitat per organitzar xerrades, cicles de conferències i activitats*)* organitzats per la comunitat Sparki Bluemix. L'assistència en aquests meetup's permetrà estendre els coneixements a partir d'altres usuaris.

Producte	Unitats	Preu Unitats	Vida útil	Amortització	Preu
Assistència a meetup's	1	25 €	-	-	25 €

Taula 11: Altres costos

1.7.2 Control de desviacions

Al final de cada història del projecte, es farà una petita valoració econòmica indicant si s'ha desenvolupat la tasca dins del pressupost establert (estimant-ne tant els recursos humans com els recursos materials) o bé s'ha excedit i, per tant, caldrà modificar el pressupost per tenir-ne un de més ajustat.

En l'apartat 6, Valoració d'alternatives i pla d'acció, es detallen les possibles alternatives a imprevistos que es puguin trobar durant el desenvolupament del projecte. A continuació es detalla, de manera breu, com afectaria aplicar l'alternativa al pressupost.

- Història 3: Es proposa utilitzar un algorisme alternatiu al k-means com ara naive Bayes. Aquest canvi afectaria, gairebé de manera ínfima, el consum de recursos al MareNostrum III.

- Història 6: En el cas de no trobar un conjunt de dades reals per al MareNostrum III es dedicarà la història 6 a la història 3. Aquesta alternativa podria variar el pressupost. Igual com en el punt anterior pot variar el consum de recursos però de manera ínfima.

- Història 7: Es proposa utilitzar una plataforma cloud alternativa a Bluemix com AWS (que ofereix les mateixes funcionalitats que Bluemix*)*. Aquesta alternativa no varia el pressupost ja que es demanaria un ajut per a estudiants o bé s'utilitzaria la capa gratuïta de recursos.

1.7.3 Costos totals

Tipus de Cost	Preu
Costos directes	9.444,71 €
Costos Indirectes	34,40 €
Altres Costos	25 €
Contingència (3%)	285,12 €
Imprevistos	476,1 €
TOTAL	10.265,33 €

Taula 12: Costos totals

1.8 Informe de sostenibilitat

A continuació es presenta la matriu de sostenibilitat del TFG (**figura 7**) i les valoracions corresponents que s'han considerat per a determinar la viabilitat del projecte.

Sostenible?	Econòmica	Social	Ambiental
Planificació	Viabilitat Econòmica	Millora en la qualitat de vida	Anàlisi de recursos
Valoració	8	9	7
24/30			

Taula 13: Matriu de sostenibilitat del TFG

1.8.1 Sostenibilitat econòmica

Per tal de determinar si el projecte és viable econòmicament, s'ha realitzat un pressupost tenint en compte els costos directes, els indirectes i d'altres costos. També s'ha establert un control de desviacions per assegurar la viabilitat econòmica.

Aquest projecte no el duu a terme un professional. És més, en la planificació s'ha explicat que es desenvolupa un procés d'aprenentatge per tal que després es pugui abordar el projecte amb èxit. Aquest mateix projecte si el fes un expert en el tema, hauria reduït les hores de treball però, en canvi, els recursos humans serien més elevats (a més experiència, més retribució).

Aquest projecte es fa en col·laboració amb el BSC i el Departament d'Arquitectura de Computadors i es participarà en l'ampliació del projecte SPARK4MN.

1.8.2 Sostenibilitat social

Les tecnologies Big Data es troben en expansió. Emmarcar aquest tema dintre d'una universitat afavoreix a fer més competitiva la universitat en l'àmbit. Els consumidors d'aquest TFG, principalment, seran investigadors i alumnes dels màsters propis de la FIB: MIRI i GEI, que utilitzaran els hands-on de Spark per al MareNostrum. Aquest TFG respon a la necessitat de crear material i millorar l'anàlisi Big Data amb Spark al MareNostrum III.

1.8.3 Sostenibilitat Ambiental

El MareNostrum té una despesa elèctrica equivalent al barri barceloní de Les Corts [8]. En qualsevol cas, a través de la realització d'aquest TFG no es reduirà la despesa elèctrica. El que sí que s'aconseguirà és que com Spark és més ràpid, en comparació amb altres frameworks (p.e.. Hadoop), s'aconsegueix un rendiment superior: més tasques per unitat de temps i, per tant, la petjada ecològica millora.

Les eines de maquinari preveuen una amortització llarga. Quan s'acabi el projecte, l'ordinador seguirà sent funcional per altres projectes. L'ordinador ha estat comprat en la Unió Europea i, per tant, s'ha pagat una taxa pel correcte reciclatge una vegada l'ordinador hagi superat la seva vida útil.

Pel que fa el programari s'ha decidit, sempre que sigui possible, utilitzar programari de lliure distribució, ja que aquest gasta menys recursos (allarga la vida útil de l'ordinador) que l'alternativa comercial i, socialment, és sostenible, perquè qualsevol persona podria fer-ne ús.

Finalment, pel que fa el material creat al llarg del TFG com el codi, estarà disponible a la plataforma Github i els hands-on i documents tindran llicència Creative Commons.

2. Framework Spark

El projecte Apache Spark neix l'any 2010 a la Universitat de Berkeley i l'any 2013 és atorgat a la fundació Apache Software Foundation i es publica amb la llicència Apache 2.0. Des de l'any 2013 fins avui el projecte ha anat madurant. Actualment compta amb, aproximadament, 500 contribuïdors.

Figura 6: Logotip d'Apache Spark

Font: http://spark.apache.org/

Apache Spark és un framework basat en el popular paradigma MapReduce. Un dels avantatges d'aquesta plataforma, n'és la velocitat. Spark, a diferència de Hadoop duu a terme les operacions a la memòria.

A la figura 2 es mostra l'execució d'una regressió logística executada amb Hadoop i amb Spark. Es pot observar com amb Spark el temps d'execució és menor, ja que amb Hadoop les dades intermèdies per realitzar els càlculs es desen en un sistema d'emmagatzematge i amb Spark, en canvi, s'emmagatzemen en memòria.

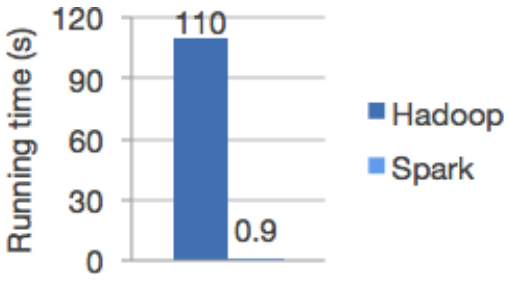

Figura 7: Regressió Logística

Font: http://spark.apache.org/

Aquest framework es pot programar en quatre llenguatges diferents de programació: Scala, Java, R i Python. Compta amb uns 80 operadors d'alt nivell que permeten treballar amb el sistema distribuït de manera transparent per l'usuari. També es compta amb shells interactives en Scala, Python i R. El framework compta amb una pila unificada (**figura 3**). És a dir, compta amb múltiples elements integrats i fa que es pugui treballar amb múltiples elements alhora amb facilitat. Spark Core, n'és el Director i s'encarrega de monitoritzar-ne les tasques (conèixer l'estat dels nodes i recollir-ne el resultat) [11].

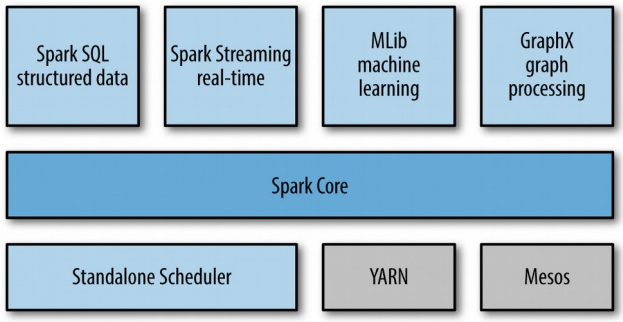

Figura 8: Pila Unificada

Font: Learning Spark – Lightning-Fast Data Analysis

(Figures 1-1. The Spark stack pàg. 3)

Els diferents components que composen Spark són els següents:

- **Spark SQL**: Aquest component permet els desenvolupadors de treballar amb bases de dades relacionals SQL.
- **Spark Streaming**: Permet processar dades en temps real. Aquest element permet per exemple que a mesura que es recullen les dades es poden analitzar.
- **Mlib**: Aquest llibreria compta amb algorismes d'Aprenentatge Automàtic (en anglès conegut com: Machine Learning*)*. Compta amb algorismes de classificació, regressió, clustering i filtratge col·laboratiu.
- **GraphX:** Aquesta llibreria permet de treballar amb grafs. Permet crear grafs dirigits en els que els nodes tenen direcció i atributs. També s'hi inclouen llibreries per poder manipular i calcular sobre el graf creat. Per exemple, s'hi inclou el graf de Page Rank, SVD++, Triangle count...
- **Standalone Scheduler, YARN i Mesos:** Aquests tres elements són Clusters Managers que

s'encarreguen de distribuir les tasques entre els diferents nodes. Existeixen tres tipus diferents de Cluster Managers.

- Standalone: aquest schedule és pensat per a petites càrregues de treball. En aquest mètode cal configurar el nombre de memòria i nombre de cores per a cada execució.
- Apache Mesos: quan arriba un nou treball, Mesos determina quins són els recursos disponibles i en retorna la informació al SparkContext, que els accepta o els rebutja. Això permet al framework de decidir si els recursos oferts són suficients per executar-ne el treball o bé s'espera a rebre una oferta amb millors recursos. Aquest model és altament escalable si la màquina escala i té més recursos El que passarà és que quan s'ofereixin recursos es podran fer millors ofertes.
- Apache Yarn: el seu scheduler treballa de manera diferent. Quan arriba una nova petició de recursos, YARN avalua els recursos disponibles i decideix quan executar-ne el treball.

Pel que fa l'emmagatzemament Spark pot utilitzar diferents sistemes d'emmagatzematge: HDFS (Hadoop Distributed File System), Cassandra, Hbase, Hive, Tachyon... tots aquests sistemes es caracteritzen per permetre treballar amb grans volums de dades.
Tal i com hem vist abans, Spark es pot programar en tres llenguatges diferents de programació i permet d'utilitzar diferents sistemes d'emmagatzemament. Aquestes característiques permeten que es pugui migrar a Spark utilitzant / mantenint els sistemes d'emmagatzemament utilitzats.

La primera versió de Spark, la versió 0.7.3, va ésser publicada al juliol de 2013. Actualment, al 2016, l'última versió 1.6.0 ha estat publicada a principis d'any. Entre la primera versió i la més nova s'han publicat un total de 16 revisions.

2.2.3 RDDs

RDD és una abstracció d'un conjunt de dades en un entorn distribuït, aquesta abstracció permet als usuaris treballar amb grans volums de dades de manera senzilla i transparent a l'usuari. Els RDD's compleixen dues característiques són elements de només lectura i tolerant a fallades.

Els RDD són tolerants a fallades. És a dir que en el cas que es perdin dades en el càlcul d'un RDD, aquest pot ser recalculat a partir de les dades originals. Aquest fet és possible ja que els RDD són només de lectura. En el sistema es desa el seguit d'operacions que es fa a cada RDD perquè en el cas que calgui recuperar dades, sigui possible. Com s'ha dit anteriorment, els RDD es distribueixen les dades en els diferents nodes que tenim al clúster. Mitjançant l'operació RDD.partiton podem configurar la granularitat de les dades. L'operació RDD.persist permet de calcular un RDD i desar-ho en memòria. Aquest fet permet que, en el cas que el programador sàpiga que les dades han de ser reutilitzades en un futur, no s'ha de calcular un nou RDD per obtenir les dades sinó que es té directament en memòria. També existeixen operacions que permeten desar RDD en el disc dur ja sigui per fer una còpia de seguretat o bé per reutilitzar el RDD en una altra execució.

Existeixen dos tipus d'operacions bàsiques: les transformacions i les accions. Les transformacions construeixen un nou RDD a partir d'un conjunt de dades o bé a partir d'un altre RDD. En són exemples les operacions: map i filter. Les operacions del segon tipus calculen un resultat a partir de les dades d'un o e diversos RDD. En són exemples les operacions count i take.

A continuació es llisten les operacions més utilitzades tant d'operacions basades en transformacions com en operacions basades en accions. El llistat complet d'operacions es troba en la guia de programació publicat per Apache Spark en la programming guide[4].

1. Transformacions

map	Permet crear un nou RDD a partir d'un altre RDD o bé a partir de dades externes. Trobem diverses operacions semblants: flatMap, mapPartitions, mapPartitions, mapPartitionsWithIndex.
filtrer	Creen un nou RDD a partir de les dades que compleixen una condició definida per l'usuari. Aquesta operació és semblant a la sentència if, totes les dades que compleixen una condició.
union	Permeten crear un nou RDD a partir de dos RDD.

4 http://spark.apache.org/docs/latest/programming-guide.html

join	Permet crear un nou RDD a partir de dos RDD les dades dels quals són *tuples* formades per tuples <key, value>. RDD1 = (K, V) i RDD2 = (K, W) es retorna RDD3 = (k, VW).
distinct	Crea un nou dataset que conté només els elements diferents de dos RDD.

2. Accions

reduce	Agrupa els valors d'un RDD a partir de les *keys* del RDD.
collect	Retorna tots els elements del RDD. Aquesta operació pot ser útil després d'executar l'operació filter que retorna un subconjunt de dades del RDD.
take	Operació semblant a collect. Aquesta permet seleccionar un nombre determinat d'elements del dataset. L'operació first retorna el primer element del dataset.
count	Permet conèixer el nombre d'elements del RDD.
saveAsTextFile	Permet escriure els continguts d'un RDD en un arxiu local. Trobem diverses operacions semblants: saveAsSequenceFile, saveAsObjectFile.
foreach	Aquesta operació permet iterar i transformar cada element contingut al RDD. Aquesta operació només es pot aplicar a RDD que tenen dades del tipus Key-Value.

2.2.2 DAG Scheduler

En aquest apartat es descriuen les diferents etapes entre que l'usuari executa un tros de codi e Spark fins que aquest acaba i rep la sortida *(*«output»*)* del codi executat.

Quan una aplicació de Spark és executada, el primer pas és crear l'objecte SparkContext. Aquest

element és una connexió lògica amb el clúster i és mitjançant aquesta referència amb la que podrem crear nous RDD. Aquest element normalment es crea en la funció main, també anomenada com Driver Program.

Els mètodes s'executen de manera "mandrosa", de l'anglès *lazy mode*. S'executen les operacions de càlcul fins que es necessiti el seu valor. Les transformacions només es calculen quan una acció en requereix el càlcul. Aquest paradigma permet incrementar el rendiment.
Per tal d'executar el programa, al clúster es crea un graf DAG (Direct Acyclic Graph) on es granula el programa principal en diferents Etapes (Stages). Aquestes etapes es divideixen en tasques (Task) que són les unitats mínimes de treball. Són les enviades als executors dels nodes del clúster. Una vegada creat el graf es transmet la informació de les diferents tasques al Cluster Manager. El Cluster Manager és l'encarregat de fer arribar les diferents tasques als nodes del clúster. Tal i com hem vist abans, pot treballar en tres modes: Standalone, Apache Mesos i Hadoop YARN.

El Cluster Manager fa arribar cadascuna de les tasques als executors localitzats en cadascun dels nodes del clúster. L'executor executa la tasca i desa la informació d'execució en memòria. En el cas que, per exemple, s'executi una operació del tipus collect, les dades emmagatzemades a l'executor es farien arribar al Driver program.

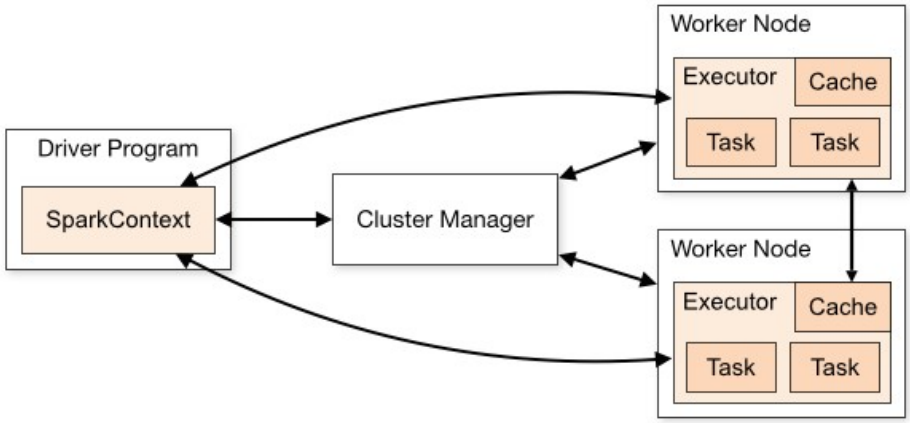

Figura 9: Arquitectura Spark

Font: http://spark.apache.org/docs/latest/cluster-overview.html

Finalment, quan tots els processos de l'Executor finalitzen, envien els resultats a l'SparkContext que fusiona els resultats en el cas que fos necessari, i mostra el resultat a l'usuari. Es pot veure el procés complet a la **figura 9**.

2.2.4 MLlib

Tal i com s'ha vist a l'apartat anterior, Spark compta amb diverses llibreries que formen part del projecte Spark. Per tant la compatibilitat d'aquestes llibreries amb el framework està assegurada. Una d'aquestes llibreries en què posarem l'èmfasi és la llibreria Mllib. Aquesta compta amb els principals algorismes d'Aprenentatge Automàtic (en anglès conegut com a *Machine Learning*).

Els algorismes d'aprenentatge Automàtic són els que són capaços d'aprendre i de millorar-ne els models i els patrons a partir de l'experiència. Aquests algorismes es poden desplegar en camps molt diversos com, per exemple, en el món de la salut, on a partir dels historials mèdics es pot determinar quins tractaments seran més eficaços per a un nou pacient, o bé estimar quin serà el consum elèctric d'un habitatge coneixent-ne l'històric de consum [12].

La llibreria Mllib és una extensa col·lecció d'utilitats i algorismes llestos per ser utilitzats. En aquest document esmentarem alguns dels algorismes amb què compta la llibreria. Es poden consultar tots els algorismes i funcionalitats de la llibreria a la guia de Mllib[5].

- **Classificació i regressió**, aquest tipus d'algorismes en basa en construir un model a partir de dades de les quals se'n coneix la categoria. Aquesta etapa s'anomena d'entrenament. Una vegada es té construït el model anterior, l'algorisme és capaç de classificar noves dades sense categoria d'acord a diverses taxonomies de l'exemple introduït. En aquesta llibreria trobem classificadors binaris: naive Bayes, regressions logístiques, arbres de decisió; classificadors amb més d'una classe: arbres de decisió i naive Bayes i algorismes de regressió.

- **Recomanadors**: Es basen en el fet de crear una recomanació a partir del model (creat a partir d'observacions) creat anteriorment. Quan l'usuari vol una recomanació es treballa amb totes les dades del model. Els recomanadors es troben estretament relacionats amb els classificadors La diferència, però, recau en què la recomanació indica els elements del model més semblants però no els classifica en una classe o en una altra. Un tipus de

5 Mllib Guide: https://spark.apache.org/docs/1.2.1/mllib-guide.html

recomandador implementat per Mllib és Collaborative filter.

- **Clustering:** és una tècnica sense supervisió. És a dir, a diferència de la Classificació, aquest tipus d'algorisme no necessita conèixer el tipus de dades sinó que el model es construeix a partir de les observacions. Un tipus d'algorismes no supervisat implementat per Mllib és k-means . [13]

2.2.5 Emmagatzemament

HDFS (*Hadoop Distributed FileSystem*), es basa en dividir els fitxers en blocs de mida fixa i desar-los de manera distribuïda en un clúster. Hadoop segueix el paradigma master-slave. El màster s'anomena NameNode i s'encarrega de regular l'accés a les dades. Les dades s'emmagatzemen en els diferents DataNode. El NameNode és l'encarregat de mantenir l'arbre de directoris. Quan un usuari vol accedir a les dades fa una consulta al NameNode i aquest li retorna en quin DataNode es troben les dades. Per tal d'augmentar la disponibilitat del sistema es podria comptar amb diversos NameNodes, on tenen la informació de l'arbre de directoris replicada i sincronitzada. Els diferents blocs de dades també poden estar replicats en diferents DataNodes[14]. A la **figura 10** es pot observar un esquema de HDFS.

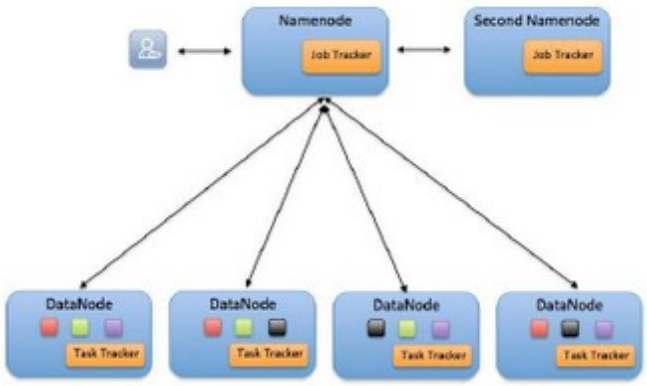

Figura 10: Esquema HDFS

Font: Cassandra File System Over Hadoop Distributed File System

Una alternativa a HDFS és CFS (*Cassandra File System*) aquest sistema de fitxers està basat en la

base de dades Cassandra. La principal diferència entre HDFS i CFS és que redueix l'overhead de consultar el NameNode i s'elimina el punt de fallada del fet que no estigui operatiu el namenode. CFS treballa de manera descentralitzada i no existeix un namenode central, sinó que tots poden servir dades (sistema P2P). Els diferents nodes s'interconnecten entre ells creant una xarxa de nodes interconnectats. **Figura 11**.

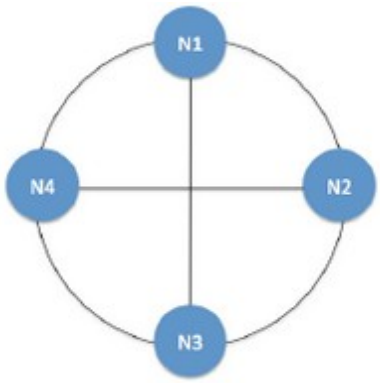

Figura 11: Esquema *CFS*

Font: Cassandra File System Over Hadoop Distributed File System

3. Realització dels experiments

En aquest apartat es descriuen les diferents infraestructures (Infraestructura física, cloud[6] privat i cloud públic) en què es realitza aquest treball de final de grau. Els resultats dels experiments es detallen en el pròxim capítol.

Infraestructura física: La infraestructura física amb què es realitzen les proves és el Supercomputador MareNostrum III. Aquest Supercomputador pertany al Barcelona Supercomputing Center. És l'ordinador més potent de l'Estat espanyol i un dels més potents d'Europa[7]. L'ordinador està instal·lat a Torre Girona al Campus Nord a la Universitat Politècnica de Catalunya.

Cloud privat, El cloud privat és una infraestructura en què els diferents nodes són lògics (la infraestructura es virtualitza[8]). Aquesta màquina pertany al Departament d'Arquitectura de Computadors de la Facultat d'Informàtica de Barcelona.

Cloud públic: El cloud públic, és una empresa privada que ofereix uns serveis (en aquest cas infraestructura Big Data) i que cobra a l'usuari en funció dels serveis utilitzats. En concret, es duran a terme un seguit d'experiments en DataBricks[9] que ofereix Spark al cloud.

3.1 Infraestructura física

El MareNostrum III es caracteritza per les altes prestacions de maquinari. És format per un total de 2.056 nodes de càlcul i una memòria de 115TB que doten la màquina amb una capacitat de càlcul d'1,1 Petaflops. Els nodes estan dotats amb processadors Sandy Bridge de 8 cores i una freqüència de 2.6 GHz. La memòria màxima amb què està dotat cada core és, com a màxim, de 8GB. L'emmagatzemament és de 2 Petabytes (1 Petabyte equival a deu elevat a quinze bytes). La comunicació entre els diferents nodes es fa mitjançant Infinband FDR10, amb una velocitat de transferència de dades de 40GB/S o bé amb Gigabit Ethernet, amb una velocitat d'1GB/S. L'ordinador treballa amb el sistema operatiu Linux - SuSe[10].[15]

6 Cloud: paraula anglesa que fa referència a *Informàtica en el núvol*: Sistema d'emmagatzematge i ús de recursos informàtics basat en el servei en xarxa, que consisteix a oferir a l'usuari un espai virtual, generalment a Internet, en què pot disposar de les versions més actualitzades de maquinari i programari. Termcat: http://www.termcat.cat/

7 Afirmació a partir de les dades del TOP500. Enllaç: http://www.top500.org/list/2015/11/

8 Virtualització: Sistema que permet compartir una màquina física per a executar diverses màquines virtuals que aprofiten els recursos lliures de processador, memòria, disc i connexió de xarxa. Termcat: http://www.termcat.cat/

9 Databricks: enllaç a la pàgina corporativa: https://databricks.com/

10 Linux Suse: Distribució destinada a entorns empresarials. Enllaç: https://www.suse.com/

Actualment, l'ordinador és destinat a tasques científiques i de recerca. Entre les quals trobem estudis sobre les ciències de la terra: prediccions meteorològiques i del canvi climàtic; estudis bioinformàtics: modelatge de molècules, estudi del genoma i proteïnes[16].

3.1.1 SPARK4MN

Spark4MN és una eina que permet executar Spark al Supercomputador MareNostrum III. L'eina treballa com un interlocutor entre Spark i la cua de processos del Supercomputador (**Figura 12**), l'eina s'encarrega de posar en cua la feina (una execució) al sistema LSF[11], inicialitzar els diferents recursos, executar l'aplicació i, finalment, s'obté un conjunt d'arxius amb informació sobre l'execució i mètriques d'aquesta.

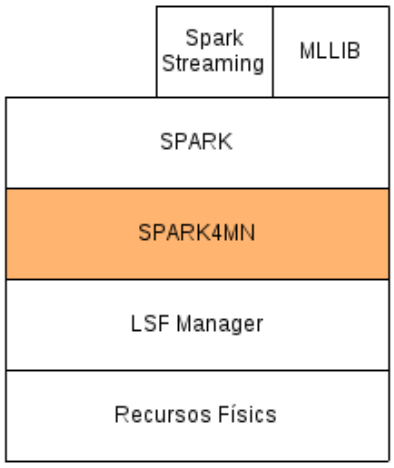

Figura 12: Spark4MN

Font: Elaboració pròpia

Per tal de treballar amb SPARK4MN cal indicar, mitjançant un arxiu de configuració, quin paquet volem executar i quina és la classe d'entrada (classe main). L'arxiu de configuració també permet seleccionar quins recursos seran utilitzats. A continuació es llisten totes les variables de l'arxiu de configuració.

- Versió de Spark
- Nombre de nodes
- Nombre de workers per node

11 LSF: Sistema que permet executar crear cua d'execucions.

- Nombre de cores per worker
- Memòria del worker
- Worker Affinity: Core, Socket, Node
- Configuració de Xarxa: Ethernet o Infiniband
- Emmagatzemament: HDFS, Cassandra

Una vegada s'executa i finalitza un workload (càrrega de treball) s'obté l'ouput del programa, mètriques de l'execució i també el nombre de recursos emprats.

- Temps d'execució del màster
- Temps d'execució dels workers [1, ... N]
- Nombre de nodes
- Nombre de workers
- Nombre de workers per node
- Nombre de cores per worker
- Memòria per worker

3.2.3 Realització dels experiments

L'accés a la màquina es realitza mitjançant el protocol SSH. Per realitzar els experiments s'utilitza l'eina SPARK4MN i l'eina de benchmarking bsc.spark disponible a Github[12]. Les eines de benchmarking, permeten mesurar i avaluar el rendiment del framework en un entorn de proves, que permeti extrapolar els resultats amb aplicacions reals. [17]

De manera local es compila l'eina mitjançant Maven que s'encarrega de baixar els binaris de Spark i totes les dependències necessàries. Després, amb l'eina SCP es va pujar l'arxiu executable a la infraestructura MareNostrum III.

Amb l'arxiu de configuració, que és necessari per poder executar les diferents execucions, es van definir les diferents variables de la màquina (per realitzar les proves): nombre de cores, nombre de memòria i tipus de connexió.

12 Bsc.spark: https://github.com/rtous/bsc.spark

Pel que fa al punt d'entrada del programa (funció main i pas de paràmetres) es defineix en aquest arxiu en la variable DOMAIN_PARAMETERS.
Les proves es van realitzar executant dos algorismes d'aprenentatge automàtic de diferent natura: k-means*:* algorisme no supervisat i naive Bayes: algorisme supervisat, que s'expliquen en profunditat en el capítol següent.

Cada test es fa tres vegades i es calcula el temps d'execució com la mitjana d'aquests tres temps. En els casos en què s'ha treballat amb un conjunt de dades grans ha estat necessari de modificar el valor de wallclock (temps màxim d'execució, per defecte fixat en quinze minuts) de l'arxiu de configuració de SPARK4MN. Per obtenir el temps d'execució es realitza de manera manual, consultant els arxius de log[13] on conté informació sobre el temps d'execució.

3.2 Cloud Privat

El cloud privat té unes prestacions menors a les de la infraestructura física. En concret consta de quatre nodes amb quatre cores. La màquina consta d'un total de 27.2 GB de memòria repartida de manera equitativa entre els seus nodes (6.8 GB per node). Pel que fa a l'emmagatzemament consta d'una unitat HDFS de 231.16 GB. Aquesta màquina virtual és d'ús exclusiu per la recerca que es realitza en el Departament d'Arquitectura de Computadors.
A diferència del Supercomputador MareNostrum III en aquesta màquina per executar les aplicacions no s'utilitza cap eina intermèdia sinó que per executar treballs amb Spark s'executa directament amb l'eina spark-submit[18].
Aquesta plataforma consta de la interfície web en què es pot veure l'estat d'execució de les diferents aplicacions a la màquina i l'historial d'execucions. També és possible veure-hi en temps real quines tasques s'estan executant mitjançant la interfície web: Spark-UI.

3.2.1 Realització dels experiments

Per realitzar els experiments s'ha utilitzat la mateixa eina de benchmark anterior. Per treballar amb aquesta màquina es treballa d'igual manera que en la màquina anterior amb els protocols SSH i SCP. El temps total d'execució s'extreu a partir de la informació disponible a la interfície web.

13 Arxiu de log: arxiu que conté informació sobre l'execució.

3.3 Cloud Públic

En un primer moment es va plantejar d'utilitzar la plataforma cloud, tal i com consta en la planificació, Bluemix[14] propietat d'IBM on es pot treballar amb serveis sota demanda. Un d'aquest serveis és crear i executar clústers amb Spark. El problema d'aquesta plataforma és que encara es troba en desenvolupament i que no permet treballar de manera estable en la plataforma. Aquesta problemàtica va ser vista al setembre de 2015.

Finalment, es va optar per realitzar proves amb Databricks. Aquesta empresa amb seu a San Francisco va ser fundada pels creadors de Spark i que dóna resposta a la demanda creixent de clients que volen treballar amb Spark ofert com a servei (PaaS)[15].

Treballar amb aquesta plataforma va ser possible ja que se'n va facilitar accés al programa Academic Databricks on es pot treballar la plataforma en un entorn dedicat a l'ensenyament i a la recerca.

Pel que fa al maquinari, la plataforma compta amb dos clústers amb diferents recursos. El primer clúster compta amb dos nodes i 12 GB de memòria. El segon, és més potent. Compta amb un total de sis nodes i 36GB de memòria.

Aquesta plataforma, a diferència de les dues anteriors, es programa mitjançant l'ús d'*Ipython notebooks*[16]. Aquesta eina permet d'executar en temps codi escrit en Python o bé Scala en un navegador web (en la plataforma *Databricks*) i veure'n el resultat en el mateix navegador.

3.3.2 Realització dels experiments

Malauradament, no s'han pogut realitzar proves de rendiment amb aquesta plataforma ja que en treballar en un clúster compartit no ha estat possible de modificar el nombre de clústers, ni de modificar l'objecte de configuració de Spark. L'única manera de realitzar aquestes proves és crear clústers privats en Amazon Web Services (AWS)[17] i, com que aquesta opció suposava un cost en el pressupost total del projecte, va ser descartada. Es va executar k-means al clúster community però pel fet explicat anteriorment en no poder tenir diferents execucions amb diferents distribucions ha estat descartada; és per això que no s'inclou en el capítol 4.

14 Bluemix: https://console.ng.bluemix.net/
15 PaaS: Platform as a Service
16 Ipython Notebook: http://ipython.org/notebook.html
17 Amazon Web Services:https://aws.amazon.com/

En el repositori Github TFG-DeploymentdeSparkalMareNostrumIII[18] s'inclouen dues carpetes Bluemix i Databricks que contenen els notebooks que s'utilitzen per dur a terme els experiments. En cas que es volgués reproduir els experiments, cal importar el notebook en Bluemix o bé en Databricks.

18 TFG-DeploymentdeSparkalMareNostrumIII: https://github.com/albertcalv/TFG-DeploymentdeSparkalMareNostrumIII

4. Experiments

En aquest apartat s'expliquen els diferents experiments duts a terme per avaluar i comparar el rendiment de la infraestructura física MareNostrum III i el cloud privat del Departament d'Arquitectura de Computadors (DAC).

En concret, s'utilitzen dos algorismes d'Aprenentatge Automàtic (de l'anglès: Machine Learning), que són inclosos a la llibreria Mllib. El primer algorisme amb què es treballa és k-means, un algorisme de Clustering no supervisat (apartat 4.1); seguidament, es fan proves amb l'algorisme naive Bayes, algoritme de classificació supervisat (apartat 4.2).

En aquest apartat es detallen els resultats dels experiments realitzats per analitzar l'escalabilitat dels algorismes. Aquestes proves es fan mitjançant l'estudi, principalment, del speedup, scaleup i sizeup.

4.1 K-means

K-means és un algorisme d'agrupament de dades no supervisat. És a dir, a priori, l'usuari no coneix com són les dades. El model de dades es construeix a partir de les observacions (model no supervisat) on s'agrupen les diferents observacions d'acord als seus atributs en K clústers. Cada mostra s'agrupa en un d'aquest clústers de manera que la distància entre la mostra i el centre del clúster (centroide) és la mínima entre els possibles clústers. [9][13].

S'utilitza la versió de k-means disponible a la llibreria Mllib versió 2.10. Aquesta versió treballa amb els paràmetres següents: maxIterations: nombre màxim d'iteracions fins que el punt convergeix en un centroide (permet d'aconseguir més precisió); inizializationMode*:* mode d'inicialització o bé a partir d'un model o de manera aleatòria; k nombre de clústers, runs nombre de vegades que s'executa l'algorisme. En el cas que s'executi més d'una instància, l'algorisme retorna aquell amb millors resultats; epsilon: distància entre un punt i el centre del clúster en què convergeix.

4.1.1 Dataset i Paràmetres

S'han definit tres datasets de diferent distribució. A la **taula 1** es poden observar els diferents datasets. El nombre de punts són les diferents mostres amb què es compta, i el nombre d'atributs o característiques d'aquesta mostra.

Aquests es generen de manera dinàmica amb la llibreria data_generator[19]. Les dades es generen i es desen directament a la memòria i, per tant, en aquest experiment.

Identificador	Punts	Atributs
1M100d	1.000.000	100
10M10d	10.000.000	10
100M1d	100.000.000	1

Taula 14: Dataset k-means

Els paràmetres d'entrada de l'algoritme són els següents: numofpoints: nombre de punts; numofcolums: nombre d'atributs o també anomenats com dimensions (es defineix el valor per cada conjunt de dades a la **Taula 14**); numofcenters: nombre de centres fixat a 5; numofiterations: nombre d'iteracions fixat a 10; num-of-trials: nombre de tests fixat a 1; inter-trial-wait: fixat a 3.

El cloud privat, té menys recursos que la Infraestructura física i, per tant, es duen a terme proves amb el màxim de recursos que el cloud privat pot oferir. Per tal de poder comparar una plataforma amb l'altra els recursos de cadascuna d'aquestes infraestructures s'ha definit a l'apartat 3.1 Infraestructura física i 3.2 Cloud Privat, Infraestructura física i cloud privat, respectivament .

Les proves es fan amb un total de 4 nodes equipats amb 1, 2 i 4 cores, equipats cadascun d'ells amb 1GB de memòria RAM. A la **Taula 2**, es poden observar les diferents arquitectures utilitzades.

Identificador	Nodes	Cores	Memòria (en total)
1 core (Seqüencial)	1	1	1GB
4 cores	4	4	4GB
8 cores	4	8	8GB
16 cores	4	16	16GB

Taula 15: Descripció de recursos

19 La llibreria està disponible a https://github.com/albertcalv/bsc.spark/blob/master/src/main/scala/bsc/spark/perf/spark/DataGenerator.scala

4.1.2 Speedup

El speedup analitza quina és la reducció de temps d'una execució utilitzant N processadors respecte a la versió seqüencial (execució amb 1 sol processador). [19]

El primer pas ha estat executar la versió seqüencial i, seguidament, hem escalat la màquina a 4, 8 i 16 cores equipats amb 1GB de memòria RAM per core. En l'execució seqüencial, executant k-means amb el dataset 1M100d, el temps d'execució és de 356,65 segons i en la infraestructura física i en el cloud privat, és de 1560 segons. En l'execució amb 4 cores (1 core per node) s'aconsegueix una reducció del temps en un 92,62% en la infraestructura física i d'un 92,69% en el cloud privat. A continuació s'inclouen els temps recollits d'execució amb els tres datasets definits anteriorment: 1M100d, 10M10d i 100M1d amb 4 cores, 8 cores i 16 cores en ambdues plataformes (Figura 1, Figura 2 i Figura 3).

En la infraestructura física, l'execució amb el dataset 1M100d (**figura 13**), el temps d'execució no mostra una millora notable en passar de treballar amb 4 a 8 cores. En canvi, de 8 a 16 cores es percep una millora de menys del 20%. En l'execució dels datasets 10M10d i 100M1d, s'observa un temps d'execució millor. En el cas de 10M10d (**figura 14**), s'aconsegueix reduir el temps d'execució en un 57% (de passar de treballar de 4 cores a 8 cores). Finalment, si escalem la màquina a 16 cores no es mostra una reducció de temps tan notable, sinó tan sols un 21% respecte a la versió de 8 cores.

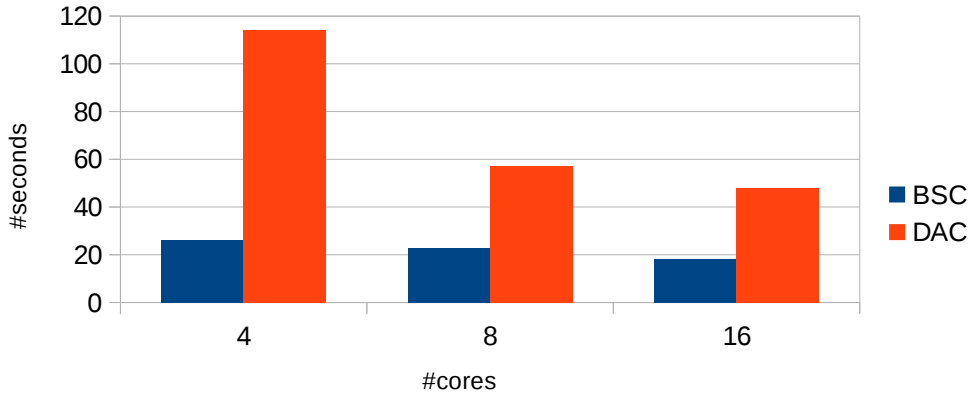

Figura 13: Temps d'execució de k-means amb el dataset 1M100d

Font: Elaboració pròpia

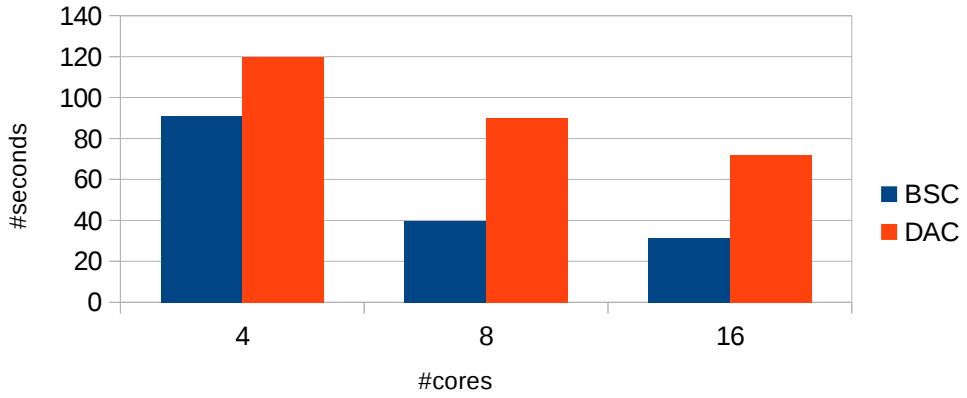

Figura 14: Temps d'execució de k-means amb el dataset 10M10d

Font: Elaboració pròpia

En l'últim cas, amb el dataset 100M1d (**figura 15**), el temps d'execució és molt més elevat que en els dos últims casos, amb 2.898 segons en comparació amb els 90,99 segons per al dataset 10M10d i 26,32 segons per el dataset 1M100d. El temps d'execució mostra bons resultats a l'hora d'escalar la màquina de 4 a 8 cores, ja que suposa una reducció del temps d'execució del 70% i quan es passa de 8 a 16, també hi ha una millora (es redueix el temps d'execució en un 51%).

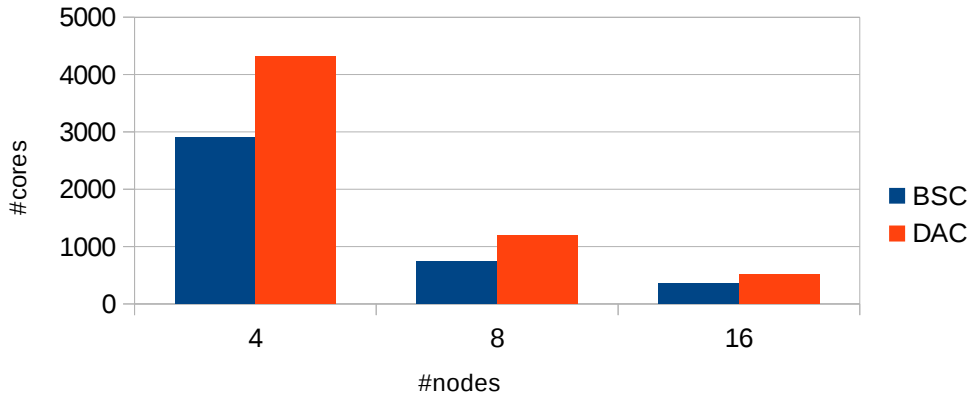

Figura 15: Temps d'execució de k-means amb el dataset 100M1d

Font: Elaboració pròpia

Pel que fa al temps d'execucions del cloud privat, són més elevats respecte a les execucions en la infraestructura física i l'escalabilitat és significativament millor. En el primer cas (1M100d) passar de 4 a 8 cores redueix el temps en un 50% i passar de 8 a 16, en un 20%. En el segon cas (10M10d) passar de 4 a 8 cores redueix el temps en un 25% i passar de 8 a 16, en un 20%. Per últim, en el dataset 100M1d passar de 4 a 8 cores, suposa una reducció molt notable, del 74%, i de 8 a 16, en un 56%.
Pel que fa al temps d'execucions del cloud privat, són més elevats respecte a les execucions en la infraestructura física i l'escalabilitat és significativament millor. En el primer cas (1M100d) passar de 4 a 8 cores redueix el temps en un 50% i passar de 8 a 16, en un 20%. En el segon cas (10M10d) passar de 4 a 8 cores redueix el temps en un 25% i passar de 8 a 16, en un 20%. Per últim, en el dataset 100M1d passar de 4 a 8 cores suposa una reducció molt notable, del 74%, i de 8 a 16, en un 56%.

S'observa que és molt més costós processar un dataset amb molts punts i poques dimensions (100M1d) que un de la mateixa grandària però amb pocs punts i moltes dimensions (1M100d). No obstant, quan escalem i afegim més recursos s'observa que el nombre de dimensions limita dràsticament l'speedup que podem aconseguir, és a dir, l'algorisme escala pitjor amb dades altament dimensionals. Això s'atribueix principalment al fet que durant l'etapa de data shuffling s'han de moure els centroides dels clústers entre els diferents recursos de còmput. A més de la necessitat de transferir dades, aquesta operació acostuma a requerir l'ús del disc, el principal responsable de la pèrdua de rendiment. Quan major és el nombre de dimensions, majors són els centroides dels clústers i major n'és l'ús del disc.

Si comparem el temps d'execució de les dues màquines, es pot comprovar com en la infraestructura privada els temps d'execució són més elevats. Aquest fet recau en què la màquina és virtualitzada i existeix un gestor de recursos. Aquest fet es tradueix en una despesa extra de recursos. També cal tenir en compte que la infraestructura física està dissenyada per treballar en entorns d'alt rendiment mentre que el cloud privat no està dissenyat per dur a terme l'alt rendiment.

Tot i les diferències entre el hardware de les plataformes es pot observar com ambdues plataformes escalen de manera semblant. En les dues plataformes s'adverteix una millora notable de passar de treballar de 4 a 8 cores, amb reduccions de temps de gairebé el 50%, i del 20% de treballar de 8 a 16 cores.

4.1.3 Scaleup

El scaleup permet analitzar com escala la màquina en augmentar de manera gradual el nombre de recursos del sistema i la mida del dataset.[19]

Per mesurar el scaleup, per tant, incrementem de manera gradual els recursos (4, 8 i 16 nodes) i la mida del dataset. S'han fet proves amb el dataset 1M100d i 10M10d. L'augment del dataset es fa de manera gradual: 1x amb 1M100d i 10M10d, 2x: amb 2M100d i 20M10d, 4x: amb 4M100d i 40M10d. A la **Figura 16** es mostra el scaleup en les dues plataformes: Infraestructura física i cloud privat.

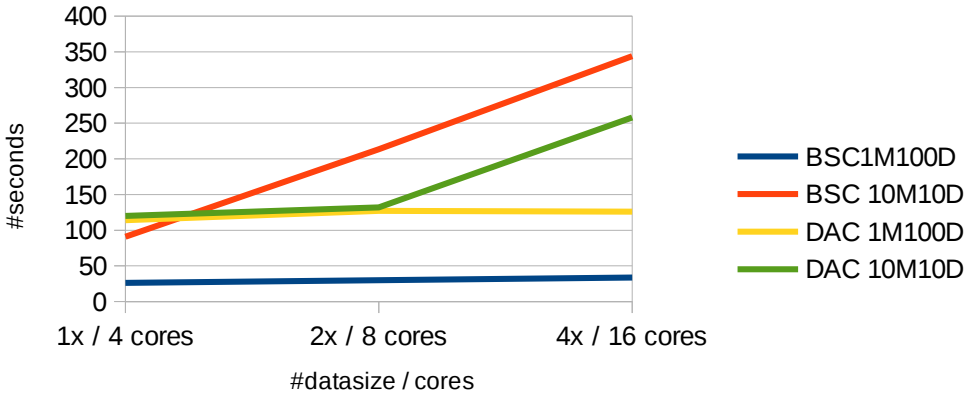

Figura 16: Scaleup BSC i DAC, k-means

Font: Elaboració pròpia

Per determinar en quins casos s'aconsegueix el scaleup ideal, ens hem de fixar en els casos en què la línia descrita amb les execucions sigui horitzontal, com és el cas de l'execució del dataset 1M100d. S'aconsegueix aquest resultat tant en la infraestructura física com en el cloud privat.

En l'altre cas 1M10d no s'aconsegueix un scaleu*p* lineal. En la plataforma del DAC, a partir de 8 nodes, el temps d'execució augmenta atesa la manca de recursos. A la màquina del BSC es pot observar com el temps amb 2x/8 cores és més elevat que l'execució amb 4x/16, ja que en el cas amb 2x/8 cores, la màquina no té prou recursos. Per tant, en aquests casos, per aconseguir uns millors resultats caldria dotar la màquina amb recursos.

4.1.4 Sizeup

El sizeup permet analitzar quin és el comportament de la màquina en augmentar de manera gradual la mida del dataset amb el qual es treballa. En aquest tipus de test es mantenen constants els recursos emprats. [19]

S'analitza el sizeup de les dues plataformes. Es duen a terme els experiments amb els datasets 1M100d i 10M10d i s'escalen de manera gradual 2x: 2M100d i 20M10d, 4x 4M100d i 40M10d. Els recursos són constants amb 8 cores o bé 16 cores equipats amb 1GB de memòria per core. A la **Figura 17,** podem observar les execucions en la infraestructura física i a la **Figura 18,** les execucions en el cloud privat.

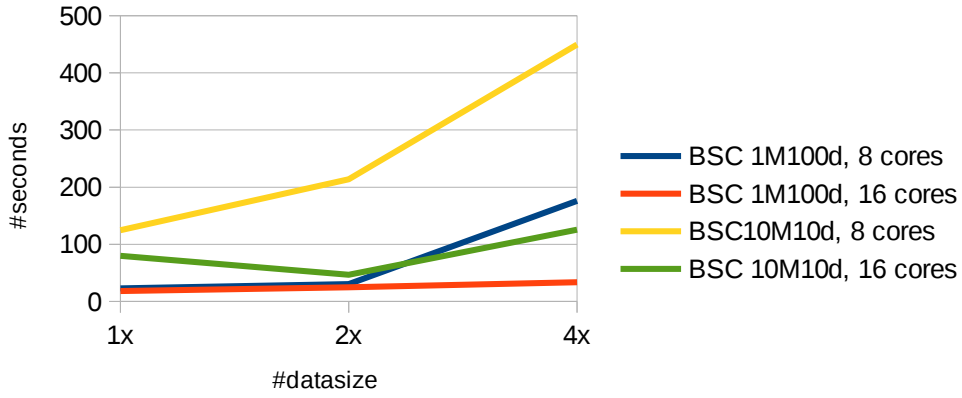

Figura 17: Sizeup en BSC, k-means

Font: Elaboració pròpia

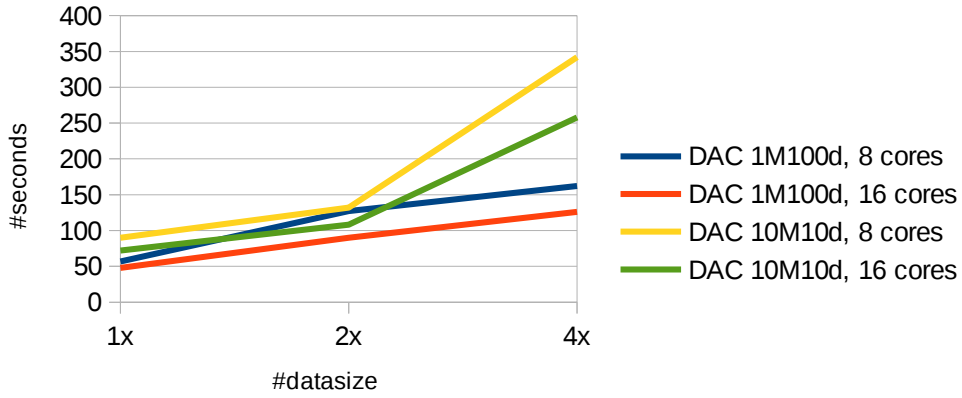

Figura 18: Sizeup en DAC, k-means

Font: Elaboració pròpia

En la infraestructura virtualitzada es pot observar que hi ha un comportament similar en les 4 execucions dutes a terme. Es pot comprovar com utilitzant un nombre més gran de recursos se'n redueixen els temps. També es percep com, a la línia que descriu les 4 execucions, en el cas 10M10d quan la mida del dataset és 4x, augmenta considerablement el temps d'execució. Aquest fet ja és perquè la màquina té pocs recursos. En el cas amb 1M100D, 4x la mida del dataset ocupa aproximadament uns 10GB.

4.1.5 Altres experiments

En aquest apartat es porten a terme experiments per estudiar quin impacte tenen els diferents paràmetres de l'algoritme k-means. Aquests experiments es fan en el cloud privat, en la infraestructura física BSC). Els resultats obtinguts tindran un comportament similar tant en una plataforma com en l'altra i, per tant, són extrapolables.

Nombre d'iteracions

El següent experiment determina quin és l'impacte d'afegir un nombre més gran d'iteracions. Aquest increment permet d'aconseguir un model més òptim. Aquesta millora es tradueix com la reducció entre la distància entre un punt de la mostra del conjunt de dades i el centroide. Aquesta distància s'anomena WSSE[20]. L'experiment ha estat dut a terme amb el dataset 10M10d.

A la **figura 19**, es veu com l'WSSE executant amb 10 i 20 iteracions és gairebé el mateix i no se'n percep una millora del model. Per aconseguir un model amb una millora notable de precisió, l'hem d'executar amb 40 iteracions, amb la qual cosa es millora l'WSSE a causa d'afegir més temps de càlcul. Finalment, l'execució amb 100 iteracions no mostra cap millora, ja que el temps de càlcul és més elevat degut al gran nombre d'iteracions.

20 WSSE: *Within Set Sum of Squared Error*

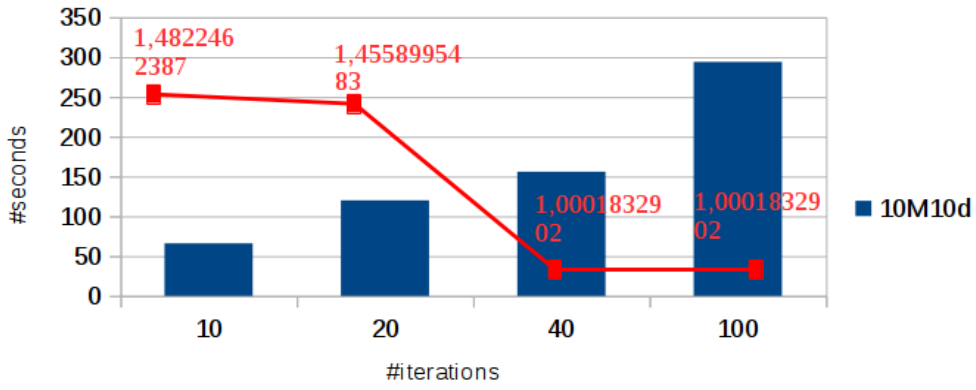

Figura 19: Millora de WSSE

Font: Elaboració pròpia

Nombre de centres

També s'ha volgut provar quin és l'impacte de treballar amb una execució de k-means amb molts centres o centroides. S'ha executat el conjunt de dades 10M10d amb diferents nombres de centres 5, 10, 20 i 40.

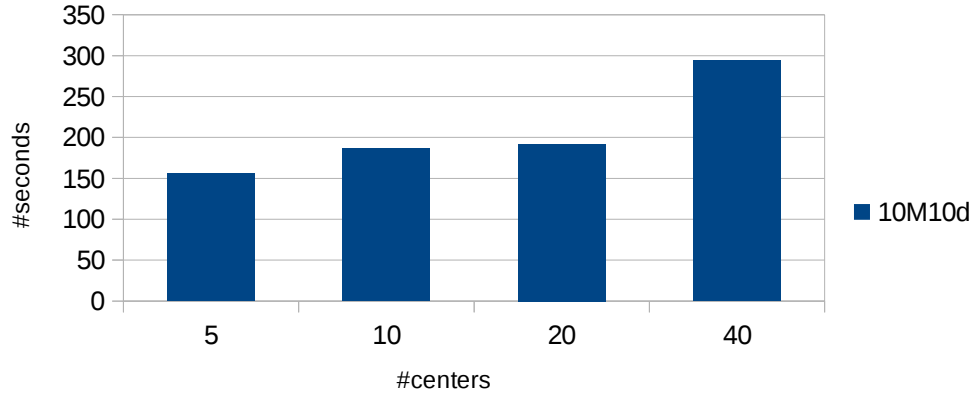

Figura 20: Millora de Centres

Font: Elaboració pròpia

Tal i com es pot veure a la **figura 20,** si augmentem el nombre de centres, el temps de càlcul és gairebé igual per a valors més petits de 20. En l'execució amb 40 centres el temps de càlcul és més gran, encara que no és el doble respecte a l'execució amb 20 iteracions.

4.1.6 Conclusions

Dels experiments anteriors podem concloure que és més costós processar un conjunt de dades amb dimensions petites i un elevat conjunt de punts, com és l'exemple del dataset 100M1d (**figura 13**). En aquest cas, hi ha més punts a la memòria i el sistema està més estressat en desar els punts en memòria. En aquest darrer cas es pot veure com hi ha una millora substancial a l'hora de dotar el sistema amb més nodes i més memòria. La millora es pot veure en les dues plataformes. Pel que fa als temps d'execució, el BSC mostra millors resultats en gairebé totes les execucions fetes. Les dues infraestructures escalen de manera semblant.

Els experiments del speedup, sizeup, scaleup s'han realitzat amb 10 iteracions. Tal i com hem observat, es pot augmentar el nombre d'iteracions. Aquest augment permetrà d'aconseguir un model més òptim, sense incrementar-ne de manera substancial el temps d'execució. Per tot el que hem vist anteriorment arribem a la conclusió que es podria treballar amb 40 iteracions.

4.2 Naive Bayes

Naive Bayes és una tècnica de classificació que crea prediccions a partir del càlcul de la probabilitat que un punt pertanyi a una classe. Aquesta tècnica està basada en el Teorema de Bayes: es descriu la probabilitat d'un esdeveniment basat en condicions que poden estar relacionades amb un altre. Aquest algorisme divideix el dataset en dos: a la primera partició de dades es duu a terme l'entrenament del model (train) i després, a la segona partició de dades, es fa la validació del model (validació). [12][13]

4.2.1 Dataset i paràmetres

Els experiments que es fan amb l'algorisme utilitzen tres datasets de la mateixa mida però de característiques diferents. En els experiments d'scaleup i sizeup s'augmenten el nombre d'exemples escalant aquest valor fins a vuit milions d'exemples. A la **taula 16** es llisten els tres datasets definits. De la taula, tenim *examples* que són el nombre total d'exemples o mostres amb què farem les proves i les features són les diferents característiques o classes que s'utilitzaran per classificar.

Identificador	Examples	Features
1ME100f	1.000.000	100
10ME10f	10.000.000	10
100ME1f	100.000.000	1

Taula 16: Dataset naive Bayes

Altres paràmetres definits en les execucions amb naive Bayes són els següents: el nombre de particions definit a 10; valor de smothing definit a 1. El conjunt de dades per a l'entrenament és del 80% i el de validació del 20%.

Els experiments es fan en la infraestructura física i el cloud de la mateixa manera que amb l'algorisme k-means. A la **taula 17** es llisten les diferents configuracions de maquinari utilitzats.

Identificador	Nodes	Cores	Memòria (en total)
1 cores (Seqüencial)	1	1	1GB
4 cores	4	4	4GB
8 cores	4	8	8GB
16 cores	4	16	16GB

Taula 17: Descripció de recursos

4.2.2 Speedup

S'analitza com millora una execució afegint més recursos de maquinari mantenint el dataset constant. El modus operandi d'aquest experiment ha estat el mateix que el seguit per executar les proves amb k-means. S'ha executat els datasets 1ME100f, 10ME10f i 100ME1f amb diferents recursos: 4 cores, 8 cores i 16 cores.

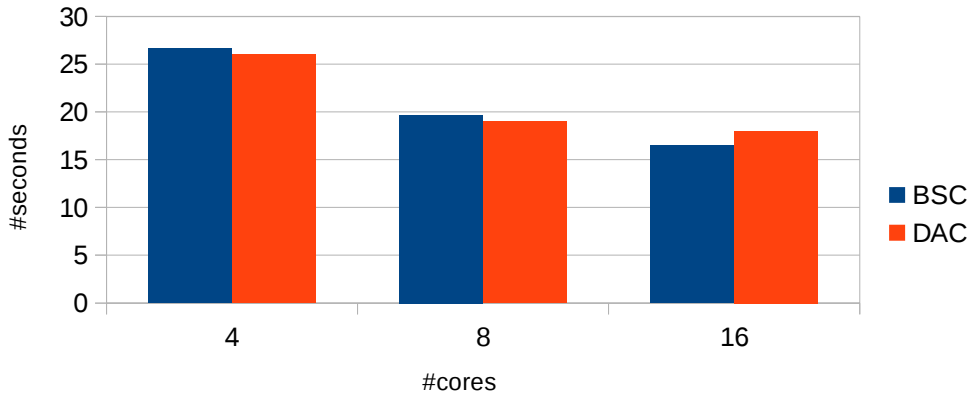

Figura 21: Temps d'execució de naive Bayes amb el dataset 1ME100f

Font: Elaboració pròpia

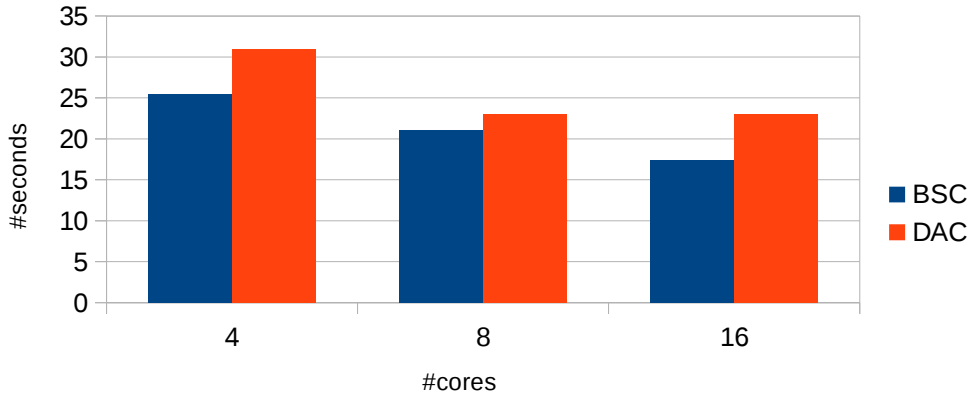

Figura 22: Temps d'execució de naive Bayes amb el dataset 10ME10f

Font: Elaboració pròpia

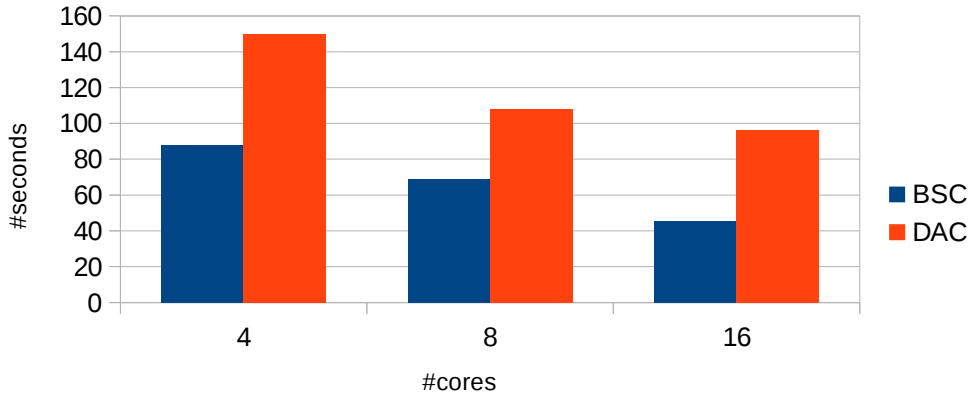

Figura 23: Temps d'execució de naive Bayes amb el dataset 100ME1f

Font: Elaboració pròpia

En el primer cas (execució 1ME100f, **Figura 21**), el temps d'execució de treballar de 4 a 8 cores en el BSC, millora un 35% i en el DAC, un 26,92%. De 8 a 16 cores, respecte a la versió amb 8 cores, millora un 16% al BSC i un 5% al DAC. Al segon experiment (execució 10ME10f, **Figura 22**), escalar de 4 a 8 cores, mostra una millora al BSC de 17,49% i al DAC del 25,80%. De 8 a 16 cores, millora un 16,97% al BSC mentre que al DAC no hi ha una millora. Finalment, l'últim experiment (execució 100ME1f, **Figura 23**), de 4 a 8 cores, es millora un 21,36% al BSC i un 34'21% al DAC. De 8 a 16 cores, es percep una millora de 28% al BSC i 11,11% al DAC.

4.2.3 Scaleup

El scaleup es fa d'una manera igual que en l'algoritme k-means, incrementant de manera progressiva la mida del dataset i el nombre de nodes. S'utilitzen els datasets 1ME100f i 10ME10f que s'escalen de manera gradual a 2x: 2ME100f, 20ME10f i 4x: 4ME100f i 40ME10f.

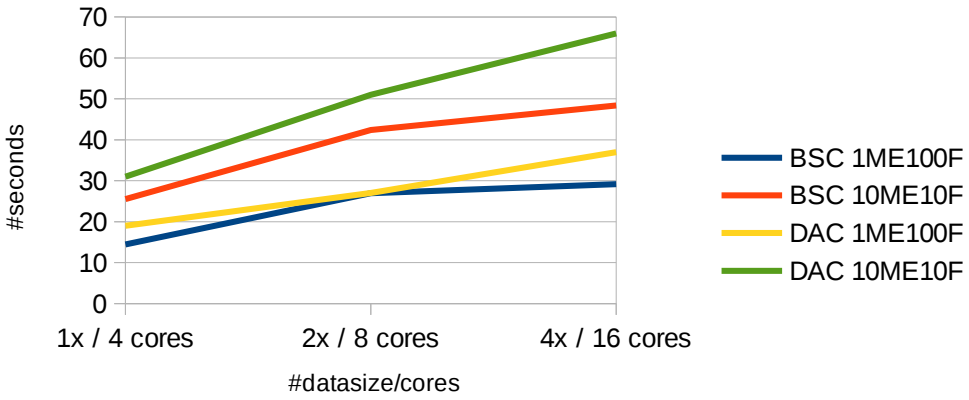

Figura 24: Scaleup en BSC i DAC, naive Bayes

Font: Elaboració pròpia

Ambdues plataformes descriuen un línia semblant (**Figura 24).** En l'execució amb el dataset 1ME100f es mostra un comportament similar. En canvi, amb el dataset 10M10f, al DAC el temps d'execucció és més gran.

4.2.4 Sizeup

En aquest apartat s'analitza el sizeup. De la mateixa manera que en l'algorisme k-means, es fan servir els datasets 1M100f i 10M10f i s'escalen de manera gradual 2x: 2M100f i 20M10f, 4x 4M100f i 40M10f. Els recursos són constants, 8 cores o bé 16 cores equipats amb 1GB de memòria per *core*.

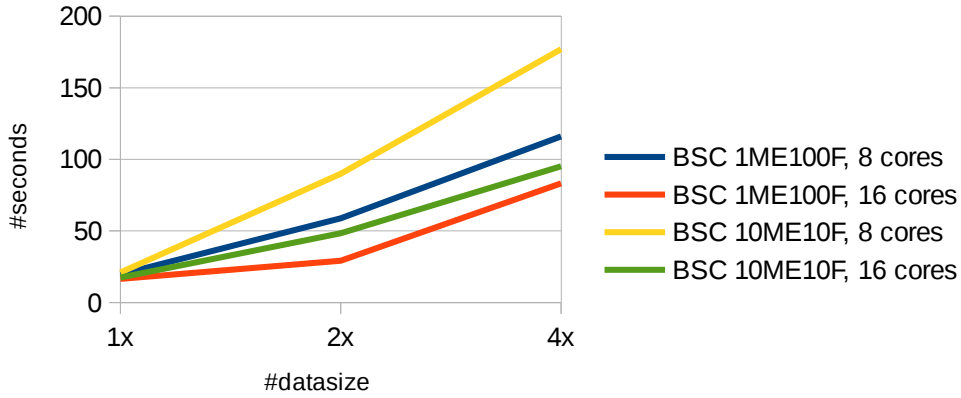

Figura 25: Sizeup en BSC, naive Bayes

Font: Elaboració pròpia

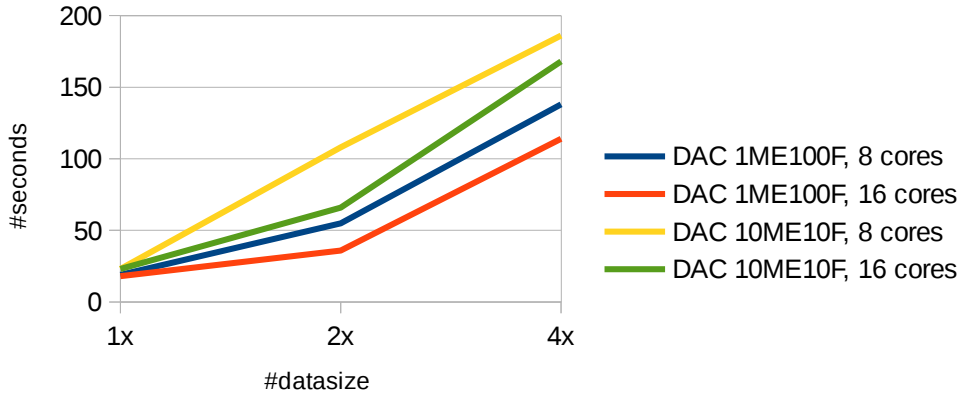

Figura 26: Sizeup en DAC, naive Bayes

Font: Elaboració pròpia

A la **figura 25** es poden observar les execucions en la infraestructura física mentre que en la **figura 26** les execucions són al cloud privat. Es pot comprovar com en les dues infraestructures, quan augmentem el datasize a 2x es dibuixa una línia gairebé horitzontal. Contràriament, quan augmentem el datasize a 4x creix de manera exponencial i no pas horitzontalment.

4.2.5 Conclusions

Tal i com s'ha vist el temps d'execució amb naive Bayes mostra uns temps d'execució més baixos que amb k-means. Aquestes mètriques no serveixen per comparar els resultats obtinguts d'un algorisme amb l'altre, ja que un algorisme és de clustering i l'altre algorisme és de classificació. És per aquest motiu que es pot observar que els temps d'execució són més baixos respecte a k-means.

Dels experiments anteriors podem concloure que de la mateixa manera que amb l'algoritme k-means és més costós processar els datasets amb moltes mostres i poques dimensions. En concret per a naive Bayes, s'evidencia que l'escalabilitat és dolenta ja que en el millor dels casos la millora és d'un 35%. En l'algorisme k-means s'han observat millores de l'ordre del 70%. Els temps d'execució són millors al BSC tot i que per als datasets 1ME100f, 1ME10f, la diferència de temps és molt petita.

En conclusió, es pot comprovar, que els dos algorismes ofereixen una escalabilitat diferent. En passar de treballar en una màquina seqüencial (1 core) a treballar amb una màquina amb 4 cores (1 core per node) es pot observar com el speedup millora magnituds més grans del 90% en els dos algoritmes analitzats. K-means ofereix millor escalabilitat en comparació amb els resultats obtinguts amb naive Bayes.

5. Impacte de partició, disc i xarxa

En aquest apartat es detallen els diferents experiments duts a terme per avaluar diferents aspectes que n'afecten el rendiment. En concret, es fan diferents proves per analitzar la partició, l'ús de disc i xarxa. Els experiments de partició s'han dut a terme tant en la infraestructura física com en el cloud privat. Els experiments de disc i de xarxa només s'han dut a terme al MareNostrum III.

5.1 Partició

La partició és un procés en què es divideix un RDD en diversos subconjunts, s'envia una o més particions a cadascun dels nodes del clúster. Es garanteix que les dades contingudes en una partició estaran en el mateix node. Per defecte, en tots els experiments hem fixat la variable partició igual a 10.

Aquest nombre de particions és configurable en el sentit que, per defecte, Spark particiona les dades d'acord al nombre de cores del clúster. En aquest experiment el que s'ha fet és dur a terme diferents execucions de l'algorisme k-means i modificar el valor de la partició per tal d'observar si recau en un temps d'execució extra [20].

Els experiments s'han dut a terme tant en el cloud públic com en la infraestructura física. Es realitzen amb el datasets 10M10D i amb 8 cores o bé amb 16 cores. Hem realitzat els experiments amb diferent valor de partició. A les dues primeres taules s'utilitzen 8 cores i es proven diferents valors de partició 1, 4, 8 i 1000 en el BSC (**taula 18**) i al DAC (**taula 19**). A la tercera i quarta taula s'utilitzen 16 cores, amb diferents valors de partició 1, 8, 16 i 1000 en les dues infraestructures BSC (**taula 20**) i al DAC (**taula 21**).

BSC	8 cores
Partition: 1	843,28
Partition: 4	69,82
Partition: 8	31,21
Partition: 1000	57,48

Taula 18: Impacte partició, 8 cores, BSC

DAC	8 cores
Partition: 1	348
Partition: 4	114
Partition: 8	90
Partition: 1000	120

Taula 19: Impacte partició, 8 cores, DAC

Partition: 1432BSC	16 cores
Partition: 8121,57 Partition: 1	34,89
Partition: 16	25,14
Partition: 1000	33,03

Taula 20: Impacte partició,16cores, BSC

DAC	16 cores
Partition: 8	90
Partition: 16	44
Partition: 1000	78

Taula 21: Impacte partició, 16 cores, DAC

S'observa un bon rendiment quan el nombre de particions és igual o més gran al nombre de cores. En el cas que la partició sigu molt gran (cas amb 1000 particions), llavors hi ha un overhead, atès que hi ha més comunicació al sistema en comunicar les diferents particions. Si la partició és més petita al nombre de cores es creen totes les dades en un mateix procés, amb la qual cosa no es pot paral·lelitzar el model.

5.2 Ús de disc

A l'apartat 4, les dades per fer els experiments es generaven de manera dinàmica. És a dir, les dades es creen directament en memòria. En aquest apartat el que es fa és generar anteriorment les dades al disc i, després, es llegeixen per tal de computar-les. Per tant en aquest experiment es vol determinar quin és el cost de llegir les dades de disc.

Per realitzar els experiments es creen els datasets 1M100d, 10M10d i 100M1d per k-means i 1M100f 10M10f i 100M1f per naive Bayes i es desen al disc. En executar l'eina de benchmarking es llegeixen les dades anteriorment creades. En aquests experiments realitzats en la infraestructura física MareNostrum III, s'utilitzen quatre nodes amb quatre cores i 16GB de memòria.

BSC - Kmeans	Dades gen. automàticament	Dades llegides de disc
1M100D	18,09	28,92
10M10D	31,53	41,13
100M1D	357,41	529,12

Taula 22: k-means amb dades llegides de disc, BSC

BSC – NaiveBayes	Dades gen. automàticament	Dades llegides de disc
1M100F	16,51	21,61
10M10F	17,56	22,90
100M1F	45,41	53,31

Taula 23: naive Bayes amb dades llegides de disc, BSC

Si comparem els temps d'execució de la segona columna, que conté els temps d'execució generant les dades automàticament en memòria amb la tercera columna, podem veure que conté els temps amb dades llegides de disc. Es pot confirmar, com en tots els casos, els temps d'execució són més alts degut a l'overhead de llegir les dades del sistema GPFS (disc) on s'han desat. Es pot observar com en k-means, menys en l'últim cas, l'overhead de temps és gairebé constant i, amb naive Bayes, també són gairebé constants l'overhead introduït en llegir de disc.

Aquest experiment no s'ha dut a terme en la infraestructura en la plataforma cloud, ja que el sistema d'emmagatzematge que s'utilitza no és el mateix que al MareNostrum III. En la plataforma cloud s'utilitza HDFS mentre que al MareNostrum III s'utilitza GPFS, per tant, no es poden comparar les dues infraestructures.

5.3 Ús de xarxa

En l'apartat 4 els experiments realitzats al MareNostrum III s'han realitzat utilitzant Infiniband que ofereix una velocitat de transferència de 40GB/s. En aquest experiment s'utilitza Ethernet que té una velocitat de transferència més baixa d'1 GB/s. Aquesta prova cerca d'esbrinar quin és l'impacte d'utilitzar Ethernet o Infiniband. Les proves es duen a terme utilitzant quatre nodes amb quatre cores cadascun i 16 GB de memòria RAM.

Dataset (10 particions)	Ethernet (1 Gb/s)	Infiniband (40 Gb/s)
1M100d	18,46	18,09
10M10d	30,87	31,03
100M1d	377,30	357,41

Taula 24: Impacte de xarxa

Es pot observar com en els dos primers casos (1M100D i 10M10D) no existeix gairebé diferència de temps. Aquesta circumstància es pot atribuir al fet que hi ha poca comunicació entre els nodes i que es duu a terme, en els dos casos, en el mateix temps. En l'últim cas, on existeix més

comunicació, es pot observar com el temps d'execució amb Infiniband és més baix respecte a l'execució utilitzant Ethernet.

6. Millores

En aquest apartat es detalla un seguit de millores amb què s'intenta optimitzar el rendiment de les execucions amb el Framework Spark. Aquestes optimitzacions són publicades a la documentació d'Oficial de Spark[21]. Aquesta documentació indica que les millores bàsiques per aconseguir un rendiment òptim recauen en el fet d'utilitzar Kryo Serialization i Memory Tunning[22].

6.1 Serialització de dades

El concepte de serialització de dades es basa a codificar els diferents objectes Java a tires de bytes amb la finalitat d'emmagatzemar-lo o bé d'enviar-lo per xarxa.[21]. Desar les dades de manera serialitzada permet estalviar memòria i també redueix l'ús de xarxa s'envien menys quantitat de bits.

Per desar les dades i comunicar-les per xarxa, Spark utilitza per defecte Java Serialization. Aquest mètode és flexible però lent[22]. Una possible alternativa és l'ús de Kryo library que serialitza els objectes en menys temps respecte a Java serialization i de manera més compacta. Aquest mètode no suporta tots les classes de Java, com sí que ofereix el primer tipus de serialització i per obtenir un millor rendiment cal enregistrar les classes que s'utilitzaran. En les primeres versions de Spark era inestable la llibreria kryo i, per això avui dia encara no s'utilitza aquest mètode per defecte, tot i que se'n recomana l'ús [22][23].

S'ha dut a terme un seguit d'execucions amb aquest mètode per observar si aporta millores en l'execució de l'algorisme k-means. Les proves es fan utilitzant quatre nodes amb quatre *cores* cadascun i 16 GB de memòria RAM.

Dataset (10 particions)	BSC (Java Serialization)	BSC (Kryo Serialization)
1M100d	18,09	18,31
10M10d	31,53	29,49
100M1d	357,41	351,50

Taula 25: Impacte de serialització

21 Spark1.4.0 Documentation Tunning Spark : http://spark.apache.org/docs/latest/tuning.html
22 Memory Tunning: Terme anglès que fa referència a personalització de dades. Per exemple

Es pot comprovar com, en el primer cas, hi ha poca comunicació entre nodes, el temps és gairebé el mateix. En els dos casos següents, en què hi ha més comunicació entre els nodes, es pot observar com el temps d'execució és més baix atès que s'utilitza kryo serialitzation.

6.2 Memory Tunning

Aquesta millora, igual com en el cas anterior, optimitza la memòria per tal de reduir-ne l'ús i no pagar l'overhead d'haver de llegir dades de disc.

Els objectes de Java són d'accés ràpid però consumeixen grans quantitats de memòria, Poden consumir de l'ordre de dos a cinc vegades de memòria, que la dada que desen té. Algunes d'aquestes raons són:

- La capçalera de l'objecte ocupa 16 bytes. Si l'objecte conté poca informació com un Integer, ocupa més espai la capçalera que la informació que conté. D'alta banda, els objectes del tipus String tenen 40 bytes de capçalera. En aquesta capçalera es desa informació sobre l'objecte, com ara la llargària [6.4]

- Els objectes del tipus HashMap i LinkedList, utilitzen punters als objectes anteriors i següents del mapa o llista. Aquest punter ocupa 8 bytes.

Alguna d'aquestes millores es basa en:

- Utilitzar arrays d'objectes i objectes primitius en contra d'utilitzar HashMaps i LinkedList. La llibreria fastutil[23] conté objectes que consumeixen menys memòria que els objectes per defecte de *Java*.

- Utilitzar identificadors numèrics i no identificadors basats en strings. La capçalera d'un objecte *integer* és més petita que la capçalera de l'objecte integer.

- Si el sistema té menys de 32GB de memòria RAM, es pot fer servir l'opció –XX: +UserCompressedOops que crea els punters de 4 bytes en comptes de 8 bytes.

D'aquestes millores que es proposen anteriorment, com que es treballa amb la llibreria Mllib i que, aquesta, segurament ja introdueix mètodes i tècniques per optimitzar la memòria utilitzada pels

23 Fastutil: http://fastutil.di.unimi.it/

objectes Java, estudiar i introduir millores en els objectes que s'utilitzen queda fora de l'abast *d'aquest TFG.* L'última millora sí que es podria provar en el cloud privat però, atès que es treballa en una màquina compartida, s'ha preferit no fer proves ja que podria fer-ne variar els resultats d'altres usuaris.

7. Creació dels hands-on

Un dels objectius d'aquest treball de final de grau és crear material didàctic que serveixi tant per a estudiants com per a persones interessades en Spark.

En concret, es duen a terme dos hands-on que són un tipus de guia que explica pas a pas com realitzar un determinat problema i encoratja que l'usuari en faci petits exercicis. També inclou les solucions als exercicis per tal que, si l'usuari quedés entrebancat, pogués continuar endavant tot consultant-ne la solució.

El primer hands-on té com a objectiu ensenyar i guiar els estudiants i investigadors a executar el framework Spark de manera senzilla. Al supercomputador MareNostrum III s'ha desenvolupat un hands-on anomenat Spark Deployment and Performance Evaluation on the MareNostrum III.

Aquest hands-on s'utilitzarà en l'assignatura de Cloud Computing del Màster Oficial MIRI (*Master in Research Informatics*) de la Facultat d'Informàtica de Barcelona. Per tant, permetrà que els alumnes duguin a terme proves i petites execucions en una màquina d'altes prestacions, com és el MareNostrum III. Atès que l'assignatura s'imparteix en anglès, el hands-on s'ha escrit en aquest idioma.

L'objectiu del segon hands-on és ensenyar com funciona la plataforma cloud Databricks i crear un exemple que permeti a l'usuari d'entendre com funciona Spark de manera interna. El hands-on desenvolupat s'anomena Inside Spark.

El segon hands-on, en canvi, s'utilitzarà en l'assignatura Cloud Computing del Màster Oficial MEI (Màster en Enginyeria Informàtica) de la Facultat d'Informàtica de Barcelona.

Els hands-on s'inclouen a l'annex 3 i 4, Spark Deployment and Performance Evaluation on the MareNostrum III i Inside Spark, respectivament. També es pot consultar en línia el primer hands-on al repositori SA-MIRI/Hands-on-11 a l'enllaç següent:

https://github.com/jorditorresBCN/SA-MIRI/tree/master/Hands-on-11.

7.1 Hands-on: Spark Deployment and Performance Evaluation on the MareNostrum III

7.1.1 Prerequisits i objectius

Aquest hands-on té uns prerequisits i cal que els usuaris que el vulguin utilitzar tinguin nocions en els aspectes següents:

- Programació en el llenguatge Scala.
- Coneixement bàsic del Shell Linux, de l'editor de text Vim, SSH i de *SCP*.
- Coneixement del gestor de dependències i compilador Maven.
- Cal que l'usuari tingui un compte i accés SSH al Supercomputador MareNostrum III.

Els objectius del hands-on són els següents:

- Ser capaç d'entendre com funciona el framework Spark i l'estructura d'un projecte amb aquesta eina.
- Ser capaç de desenvolupar un petit projecte amb el framework Spark, tant per crear el codi font com per incloure en el projecte totes les llibreries i dependències necessàries.
- Ser capaç d'executar un algorisme de la llibreria Mllib.

Aproximadament, s'estima que la durada del hands-on és de dues hores.

7.1.2 Desenvolupament

L'estructura del hands-on queda dividida en les seccions següents, de les quals a continuació s'inclou una breu descripció.

1. Hands-on description

    Descripció: petita introducció del contingut del hands-on i dels prerequisits necessaris per dur-lo a terme.

2. Project Structure

Descripció: aquest apartat explica quina és l'estructura bàsica d'un projecte amb Maven. S'explica com fer un petit programa per tal que en l'apartat següent es pugui comprovar si funciona correctament Maven, ja que pot donar errors de compilació. En concret, s'anima l'usuari a escriure un programa que llegeixi un arxiu de text i que compti el nombre de paraules que hi apareixen.

3. Compilation with Maven

Descripció: Aquest apartat explica les comandes per compilar i crear el paquet jar.

4. Upload the JAR using scp

Descripció: En aquest apartat s'explica com transferir el paquet .jar creat en l'apartat anterior al MareNostrum III mitjançant l'eina SCP.

5. First procedures into MareNostrum III

Descripció: S'expliquen els passos necessaris per executar un programa en Spark al MareNostrum III. Es descriu l'eina SPAR4MN: les comandes per carregar l'eina i l'arxiu de configuració necessari per poder executar el framework Spark.

6. Trace back of execution

Descripció: Una vegada s'executa un programa amb l'eina SPARK4MN es crea un seguit de fitxers que contenen un resum dels recursos utilitzats, temps d'execució, etc. En aquest apartat s'expliquen els diferents arxius.

7. K-means using Mlib library

Descripció: Aquest apartat explica com executar el popular algorisme d'aprenentatge automàtic k-means, de la llibreria Mllib.

8. Spark Benchmarking

Descripció: S'introdueix i s'explica l'eina de benchmarking bsc.spark. Aquesta eina pot ser utilitzada en el cas que l'usuari vulgui fer unes altres proves en la màquina.

7.1.3 Punts crítics

En aquest apartat es defineixen quins són el punts crítics, aquells que poden ser més conflictius i amb més dificultat.

- Punt crític 1: Compilació amb Maven. Poden sorgir errors a l'hora d'instal·lar-lo i en crear l'arxiu XML amb les dependències necessàries.
- Punt crític 2: Creació de l'arxiu .conf. Aquest arxiu és l'encarregat d'executar el .jar al MareNostrumIII. L'arxiu és sensible a majúscules, minúscules i espais i, per tant, cal tenir especial cura que la sintaxi sigui correcta.
- Punt crític 3: En el cas que l'usuari tingui poca experiència amb Scala o que sigui novell amb Spark, es pot trobar perdut amb la programació en aquest llenguatge.

Una de les possibles millores que es podrien dur a terme seria dividir el hands-on en dos. El primer hands-on contindria tota la part d'introducció (compilació, descripció de la plataforma, etc.) mentre que el segon es dedicaria exclusivament a la llibreria d'aprenentatge automàtic Mllib i analitzaria en profunditat com funciona algun dels algorismes inclosos en la llibreria com, per exemple, k-means (inclòs en el hands-on desenvolupat).

7.2 Hands-on: Inside Spark

7.2.1 Prerequisits i objectius

En aquest hands-on, igual com en el cas anterior, cal que l'usuari tingui els prerequisits següents:

- Accés a la plataforma Academic Databricks
- Programació en Phython

Els objectius del hands-on són els següents:

- Aprendre operacions/funcions de Spark
- Ser capaç de crear i de treballar amb un notebook
- Entendre com funciona Spark internament

Igual com en el hands-on anterior, la durada s'estima en dues hores.

7.2.2 Desenvolupament

Seguidament, s'hi inclou les diferents seccions que conté i una breu descripció del contingut. L'estructura del hands-on queda dividida en els punts següents.

1. Hands-on description

    Descripció: petita introducció del contingut del hands-on.

2. Databricks environment

    Descripció: s'explica com funciona la plataforma cloud Databricks i com crear un notebook per treballar. A més, també s'explica com importar el notebook creat anteriorment per dur a terme el hands-on.

3. How Sparks Internally works

    Descripció: en aquest apartat s'explica com funciona de manera interna el framework. És a dir, com es duen a terme les comunicacions entre nodes i l'assignació de tasques.

4. First program

    Descripció: s'executa l'activitat de contar les paraules del llibre *La volta al món en 80 dies.* Es fa un filtratge que cerca quines ciutats apareixen al llibre i, finalment, es visualitzen les imatges en un mapamundi mitjançat la llibreria Basemap. Durant aquest apartat es descriu l'eina que ofereix Spark per visualitzar l'estat de l'execució (Spark-UI).

5. References

    Descripció: S'inclouen referències i pàgines d'interès.

7.2.3 Punts crítics

El hands-on té poca dificultat tècnica. Gairebé tots els apartats es basen a entendre els conceptes que es proposen. L'única problemàtica possible és la instal·lació de la llibreria Basemap i, per això, es proposa l'activitat de visualitzar les ciutats en un mapamundi com a opcional.

8. Conclusions

En aquest apartat es detallen les diferents conclusions que es treuen, fruit de dur a terme aquest treball de final de grau. Primer de tot s'hi inclouen les conclusions tècniques, les conclusions personals, el treball futur; les possibles ampliacions per a aquest projecte i, finalment, la planificació i la gestió econòmica final del projecte.

Abans de tot, cal dir que s'han acomplert els diferents objectius plantejats en aquest projecte: 1. Dur a terme proves de rendiment a la plataforma MareNostrum III, 2. Comparar la plataforma MareNostrum III amb plataformes cloud, 3. Crear material didàctic.

El primer objectiu s'ha acomplert. S'han dut a terme fet diverses proves de rendiment, en concret: speedup, scaleup, sizeup i d'altres proves en què els resultats han estat inclosos en els capítols 4, 5 i 6. Durant el desenvolupament d'aquestes proves, de manera paral·lela s'han dut a terme els mateixos experiments al cloud privat(sempre que ha estat possible) per tal de comparar les plataformes. En un principi, aquest projecte volia comparar la plataforma MareNostrum III amb una plataforma cloud tal i com és Bluemix o Databricks, però com que la plataforma encara estava en beta o la impossibilitat de dur aquests experiments en la segona plataforma, van comportar que la comparativa es fes amb el cloud privat del Departament d'Arquitectura de Computadors.

Finalment, l'últim objectiu s'ha desenvolupat correctament. Dels hands-on duts a terme, cal dir que el primer ja ha estat introduït i treballat per part dels alumnes de l'assignatura Supercomputers Architecture del màster MIRI. Per tant, podem concloure que aquest hands-on ha tingut èxit. El segon hands-on desenvolupat encara no ha estat introduït en l'assignatura Cloud Computing del màster MEI. Probablement s'introduirà a partir del pròxim quadrimestre de la tardor de 2016.

8.1 Conclusions tècniques

Després de dur a terme aquest treball de final de grau, les conclusions que cal remarcar són:

- S'ha treballat amb Spark amb el llenguatge de programació Scala (programació per al MareNostrum III i DAC) i també en python (programació durant l'execució dels hands-on). Podem concloure que Spark és un framework que ofereix una gran compatibilitat i que permet utilitzar diversos llenguatges de programació.

- Hem treballat en tres infraestructures ben diferents tot reutilitzant el codi d'una infraestructura en una altra i, per tant, Spark permet portar codi de diferents llocs d'una manera senzilla.

-Per fer les proves amb l'ús de disc ha estat necessari ampliar el repositori bsc.spark. Aquesta ampliació ha estat inclosa en el paquet disponible a github bsc.spark.

- Dels resultats obtinguts, la conclusió és que depèn de quin algorisme s'utilitzi es pot obtenir una millor escalabilitat. En concret, dels dos algorismes analitzats k-means ofereix una millor escalabilitat. Cal dir també que, en passar de treballar amb una màquina seqüencial a una màquina distribuïda, les millores són d'un 90%.

- En l'apartat d'impacte de disc, xarxa i partició s'han vist algunes de les parts que afegeixen overhead a les execucions fetes.

- En l'apartat Millores sembla que, en general, es milloren bastant les execucions fent servir kryo serialization. Les altres millores no s'han pogut provar ja que modifiquen el clúster de Spark i poden afectar altres usuaris del Departament d'Arquitectura de Computadors.

8.2 Conclusions personals

Aquest treball de final de grau m'ha servit per endinsar-me en el món de Big Data i aprendre una eina tan potent com és Spark. Espero, i estic segur, que tota l'experiència obtinguda em seran útils per desenvolupar projectes amb aquesta eina.

Pel que fa el desenvolupament del projecte, he de dir que mai no havia treballat amb la metodologia scrum. Crec que aquesta tècnica ha estat important per a la finalització i el constant desenvolupament del projecte. El fet de tenir històries descrites m'ha permès de treballar en altres històries quan la història no ha pogut ser desenvolupada.

Finalment, crec que tots els resultats obtinguts són prou interessants, sobretot la part en què es fa la comparació entre plataformes. Aquests resultats poden ser útils per a d'altres usuaris. També estic satisfet del resultat final dels hands-on perquè serviran de suport educatiu per a alumnes dels màsters.

8.3 Treball futur

A priori, no es té constància que aquest projecte tingui continuïtat. A continuació es llisten diverses idees/punts de possibles ampliacions del projecte.

- Realitzar la comparativa amb una infraestructura cloud pública, d'aquesta manera es tindria una comparativa amb tres infraestructures diferents.

- Durant aquest projecte hem treballat amb dos algorismes d'aprenentatge automàtics. Pot ser interessant de fer treballar amb altres algorismes.

- En l'apartat 7 s'introdueix un seguit de punts que poden millorar el rendiment de la màquina. Seria interessant de desenvolupar-los.

- Durant aquest projecte es desenvolupen dos hands-on amb la finalitat de ser introduïts en assignatures de màster. Una possible ampliació és crear més material que faci ús d'aquestes eines d'anàlisi de dades massives.

8.4 Planificació temporal i econòmica final

D'acord amb la planificació inicial, el projecte es va planificar per tenir-lo acabat cap a principis de gener (05/01/2015) però s'ha optat per allargar-ne el termini.

La nova data de lliurament del projecte queda fixada per al 29/03/2015. En concret, es va aplicar la alternativa proposada a la història 7 a l'apartat 1.6 Valoració d'alternatives i pla d'acció, que consistia a treballar amb una plataforma cloud alternativa a Bluemix. Es va investigar amb altres plataformes *AWS* i DataBricks però sense èxit i, finalment es va facilitar la possibilitat de dur a terme les proves al Departament d'Arquitectura de Computadors (DAC). Aquesta extensió de temps ha permès, finalment, de trobar una plataforma cloud on treballar i finalitzar els experiments dintre del nou termini.

La **taula 26** inclou les dates previstes i les dates finals en què s'han dut a terme els diferents blocs.

Bloc	Descripció	Planificació prevista	Planificació final
Bloc 0	Familiarització	01/08/15 – 15/09/15	01/08/15 – 15/09/15
Bloc I	Curs de GEP.	15/09/15 – 16/10/15	15/09/15 – 16/10/15
Bloc II	Desenvolupament del projecte	15/09/15 – 19/01/15	15/09/15 – 22/03/15
Bloc III	Preparació de la defensa	05/01/15 – 20/01/16	28/03/15 – 11/04/16

Taula 26: Planificació final

El bloc II, s'ha dut a terme mitjançant scrum, a l'**annex 2** s'inclou el diagrama de Gantt final i es detallen els temps d'execució de cadascuna de les històries. Com s'ha explicat al principi, al pla d'acció, l'ús d'aquesta alternativa ha estat pactada amb el director del projecte que m'ha ajudat a trobar una plataforma alternativa.

Finalment, l'última de les històries reservada a les ampliacions s'ha fet en ampliar la història 7: Comparativa de plataformes i la història 5: hands-on (plataforma cloud).

Pel que fa a la planificació econòmica, no ha sofert cap desviació respecte a la plantejada al principi d'aquest document. Al principi es va plantejar l'ús de Bluemix però atesos els problemes vistos durant el desenvolupament d'aquest document es va abandonar la possibilitat d'ús d'aquesta eina. Aquesta va ser substituïda pel cloud del Departament d'Arquitectura de Computadors que no ha suposat un cost extra igual com en el cas de l'ús de la plataforma DataBricks per dur a terme el segon hands-on. En la següent taula s'inclouen els recursos de programari afegits durant el projecte i que, per tant, no figuren en la taula 8, Costos directes de programari.

Producte	Unitats	Preu Unitats	Vida útil	Amortització	Preu
Plataforma cloud DAC	1	0€	-	-	0€
Plataforma cloud DataBricks	1	0€	-	-	0€
TOTAL					0€

Taula 27: Costos directes de programari afegits durant el projecte

9. Índex de figures

10. Índex de taules

11. Referències

Tots els enllaços han estat comprovats el dia 15/04/2016.

[1] Jones, M. D. High Performance computing (2011). *Introduction to High Performance Computing*. Enllaç: https://www.nitrd.gov/apps/hecportal/index.phptitle=Introduction_to_High_Performance_Computing,_presentation_by_M.D._Jones,_University_of_Buffalo

[2] Schroeck, Michael & Shockle, Rebeca ,Shockle & Smart, Janet & Romero-Morales, Dolores & Tufano, Peter (2012). *Analytics: el uso de big data en el mundo real*. Enllaç: http://www-05.ibm.com/services/es/gbs/consulting/pdf/El_uso_de_Big_Data_en_el_mundo_real.pdf

[3] Tripiana, Carlos (2015, April). SPARK4MN. PATC Courses. Enllaç: http://www.bsc.es/support/PATC-MareNostrum3/2015/1stDAY/11:00-Spark4MN_A_Big_Data_Use_Case.pdf

[4] Datamation (2015, April). *Why Big Data And The Internet of Things Are A Perfect Match* Enllaç: http://www.datamation.com/applications/why-big-data-and-the-internet-of-things-are-a-perfect-match.html

[5] Gartner Staff (2016) *IT Glossary.* Enllaç:http://www.gartner.com/it-glossary/big-data

[6] Sathi, Arvind (2012) *Big Data Analytics: Disruptive Technologies for changing the game*. MC Press ISBN: 978-1-58347-380-1

[7] Tous, Ruben & Torres, Jordi & Ayguadé. Eduard (April, 2015) Multimedia Big Data Computing for In-depth Event Analysis. BigMM 2015. Beijing, China. Enllaç: http://personals.ac.upc.edu/rtous/publications/conf_bigmm15.pdf

[8] Carrera, David (2014) *Example of Challenge in Big Data, CPD Slides*. Facultat d'Informàtica de Catalunya.

[9] Herbster, Mark(2014, September). *K-Means algorithm*. Enllaç: http://www0.cs.ucl.ac.uk/staff/M.Herbster/GI07/week3/kmeans.pdf

[10] Tous, Ruben & Gounaris,Anastasios & Tripiana, Carlos & Torres, Jordi & Girona, Sergi & Ayguadé, Eduard & Labarta, Jesús & Becerra, Yolanda & Carrera, David & Valero, Mateo. *Spark Deployment and Performance Evaluation on the MareNostrum Supercomputer* (2015, April) IEEE BigData Santa Clara, CA, USA

[11] Karau, Holden & Konwinski, Andry & Wendell, Patrick & Zaharia, Matei (2015, February). *Learning Spark – Lighthing-Fast Data* Analysis. O'Really Media. ISBN 9781449358624

[12] Tom M. Mitchell (1997). *Machine learning.* McGraw-Hill Science. ISBN 0070428077

[13] Benítez, Raul & Escudero, Gerard & Kanaan, Samir. Inteligencia artificial avanzada, FUOC. Fundació per a la Universitat Oberta de Catalunya.

Enllaç: http://archivos.inteligencia-artificial.net/archivos/Raul%20Benitez%20y%20otros.%20Inteligencia%20Artificial%20Aumentada.pdf

[14] A. Mutha, Ashish & M. Deshmukh, Vaishali (2014, March), *Cassandra File System Oveer Hadoop Distributed File System*. International Journal on Recent and Innovation Trends in Computing and Communication Volume:2 Issue:3 ISSN: 2321-8169634-637

Enllaç: http://www.ijritcc.org/download/Cassandra%20File%20System%20Over%20Hadoop%20Distributed%20File%20System.pdf

[15] BSC Staff (2015, December)*MareNostrum III User's Guide*
Enllaç: https://www.bsc.es/support/MareNostrum3-ug.pdf

[16] BSC Staff (2016) *Computer Sciences* Enllaç: https://www.bsc.es/computer-sciences

[17] Bouckaert, Stefan & Vanhie-Van Gerwen, Jono, Moerman, Ingrid (2011, August) *Benchmarking computer and computer networks. FIRE Architecture media.* Enllaç: http://www.ict-fire.eu/uploads/media/Whitepaperonbenchmarking_V2.pdf

[18] Spark comunity (2016, January) Spark *1.6.1 Documentation.* Enllaç: http://spark.apache.org/docs/latest/submitting-applications.html

[19] Ayguadé, Eduard & Ramon Herrero, Josep & Jiménez, Daniel (2013) Parallelism(PAR) Analysis of parallel applications SLIDES.

[20] Miller, Heather (2012) Parallel Programming and Data Analysis. École Polytechnique Fédérale de Lausanne. Enllaç: http://heather.miller.am/teaching/cs212/slides/week20.pdf

[21] García de Jalón, Javier & Ignacio Rodríguez, José & Mingo, Iñigo, & Aitor Imaz, Aitor & Brazález, Alfonso & Larzabal, Alberto & Calleja, Jesús & García, Jons (2000). *Aprenda Java como si estuviera en primero.* TECNUN.
Enllaç: http://www4.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf

[22] Denault, Alexandre (2014) *Socket and Serialization Comp-303: Programming Techniques Lecture 12*. McGill University. Enllaç: http://www.adinfo.qc.ca/pdf/teaching/cs303/lecture12.pdf

[23] Ryza, Sandy (2015) *How-to: Tune Your Apache Spark Jobs*. Cloudera Engineering Blog. Enllaç: http://blog.cloudera.com/blog/2015/03/how-to-tune-your-apache-spark-jobs-part-1/